Hamburg | Behörde für Schule, Familie und Berufsbildung

Schriftliche Abiturprüfung
Schuljahr 2025/2026

Informatik
auf grundlegendem Anforderungsniveau

an allgemeinbildenden und beruflichen gymnasialen Oberstufen

Haupttermin
Mittwoch, 22. April 2026, 09:00 Uhr

Unterlagen für die Lehrkräfte

Diese Unterlagen sind nicht für die Prüflinge bestimmt.

Diese Unterlagen enthalten:

1. Allgemeines[1]

- Weisen Sie bitte die Prüflinge auf die allgemeinen Arbeitshinweise hin.
- Die Arbeitszeit beträgt **255 Minuten**.
- Eine gesonderte Lese- oder Auswahlzeit wird nicht gewährt.
- Hilfsmittel: Taschenrechner (nicht programmierbar, nicht grafikfähig), Rechtschreibwörterbuch

[1] Entsprechend der „Richtlinie über die Gewährung von Erleichterungen für neu zugewanderte Schülerinnen, Schüler und Prüflinge bei Sprachschwierigkeiten in der deutschen Sprache“ (MBlSchul Nr. 08, 7. Oktober 2016, S. 60) werden für die betroffenen Prüflinge die folgenden Erleichterungen gewährt:
- Die Bearbeitungszeit wird um 30 Minuten auf **285 Minuten** erhöht.
- Ein nicht-elektronisches Wörterbuch Deutsch – Herkunftssprache / Herkunftssprache – Deutsch wird bereitgestellt.

2. Rückmeldebogen für die Zweitdurchsicht

Bitte umgehend ausfüllen und

a) an die Schule senden, die die externe Zweitdurchsicht übernimmt,

oder ggf.

b) an die Kollegin/den Kollegen geben, die/der die interne Zweitdurchsicht übernimmt.

Information für die Zweitdurchsicht

Fach:
Informatik
auf grundlegendem Anforderungsniveau Kurs-Nummer: _____

Bearbeitet wurden die folgenden Aufgaben:

Aufgaben-Nr.		Anzahl	
I	von		Prüflingen
II	von		Prüflingen
III	von		Prüflingen

3. Hinweise zu den Aufgaben

- Die Lehrkraft erhält **drei** Aufgaben und reicht diese an die Prüflinge weiter.
- Die Prüflinge wählen **eine** der Aufgaben II oder III aus und bearbeiten diese.
- Sie müssen die Aufgabe I bearbeiten.
- Sie vermerken auf dem Deckblatt und der Reinschrift, welche Aufgaben sie bearbeitet haben.

4. Hinweise zum Korrekturverfahren

- Die Korrekturen werden gemäß der „Richtlinie für die Aufgabenstellung und Bewertung der Leistungen in der Abiturprüfung“ (Abiturrichtlinie) in der geltenden Fassung vorgenommen.
- Die für das Fach zuständige Lehrkraft begutachtet die Arbeiten unter Beachtung zentraler Bewertungsvorgaben und unter Kennzeichnung ihrer Vorzüge und Mängel, der richtigen Lösungen und der Fehler und bewertet jede Arbeit mit einer Punktzahl (siehe Erstkorrekturbogen). Entwürfe können ergänzend zur Bewertung herangezogen werden.
- Jede Arbeit wird sodann von der zweiten Fachlehrkraft durchgesehen, die sich entweder (auf dem Erstkorrekturbogen) der Bewertung durch die für das Fach zuständige Lehrkraft anschließt oder (auf dem Zweitdurchsichtbogen) ein ergänzendes Gutachten mit Bewertung anfertigt.
- Die oder der Vorsitzende des Fachprüfungsausschusses legt die endgültige Punktzahl fest. Beträgt die Differenz der im Erstgutachten und im ergänzenden Gutachten erteilten Punktzahlen nicht mehr als drei Punkte, bildet sie oder er den Mittelwert beider Punktzahlen; eine gebrochene Zahl wird zur nächsten vollen Punktzahl aufgerundet. Beträgt die Differenz der im Erstgutachten und im ergänzenden Gutachten erteilten Punktzahlen mehr als drei Punkte, legt die oder der Vorsitzende des Fachprüfungsausschusses die endgültige Punktzahl in Auseinandersetzung mit den erstellten Gutachten entsprechend dem Erfordernis der Einheitlichkeit und Vergleichbarkeit der Bewertung der Prüfungsleistungen fest und begründet die Entscheidung auf dem Formular „Festlegung der Note bei einer Differenz der im Erstgutachten und im ergänzenden Gutachten erteilten Punktzahlen von mehr als drei Notenpunkten“.
- Die Bewertungsbögen sind als Download zu finden auf der Plattform Prüfungen (vormals HERA) in LMS.

5. Bewertung und Erwartungshorizont

I. Bewertung

Jeder Aufgabe sind 50 Bewertungseinheiten (BE) zugeordnet. In allen Teilaufgaben werden nur ganze BE vergeben. Insgesamt sind 100 BE erreichbar. Bei der Festlegung von Notenpunkten gilt die folgende Tabelle:

Erbrachte Leistung (in BE)	Notenpunkte
≥ 95	15
≥ 90	14
≥ 85	13
≥ 80	12
≥ 75	11
≥ 70	10
≥ 65	9
≥ 60	8

Erbrachte Leistung (in BE)	Notenpunkte
≥ 55	7
≥ 50	6
≥ 45	5
≥ 40	4
≥ 33	3
≥ 27	2
≥ 20	1
< 20	0

Für die Erteilung der **Note gut** (11 Punkte) ist mindestens erforderlich, dass die Prüflinge 75 % der erwarteten Gesamtleistung erbracht haben. Dabei muss die Prüfungsleistung in ihrer Gliederung, in der Gedankenführung, in der Anwendung fachmethodischer Verfahren sowie in der fachsprachlichen Artikulation den Anforderungen voll entsprechen.

Für die Erteilung der **Note ausreichend** (5 Punkte) ist mindestens erforderlich, dass die Prüflinge mindestens 45 % der erwarteten Gesamtleistung und über den Anforderungsbereich I hinaus Leistungen in einem weiteren Anforderungsbereich erbracht haben.

Die zwei voneinander unabhängigen Aufgaben der Prüfungsaufgabe werden jeweils mit 50 Bewertungseinheiten bewertet. Die erbrachte Gesamtleistung ergibt sich aus der Summe der Bewertungseinheiten in den beiden Aufgaben.

Schwerwiegende und gehäufte Verstöße gegen die sprachliche Richtigkeit oder gegen die äußere Form führen zu einem Abzug von bis zu zwei Punkten in einfacher Wertung. Mangelhafte Gliederung, Fehler in der Fachsprache, Ungenauigkeiten in Zeichnungen oder unzureichende oder falsche Bezüge zwischen Zeichnungen und Text sind als fachliche Fehler zu werten. Ein Abzug für Verstöße gegen die sprachliche Richtigkeit soll nicht erfolgen, wenn diese bereits Gegenstand der fachspezifischen Bewertungsvorgaben sind.

II. Erwartungshorizont

Bei den auf den folgenden Seiten dargestellten erwarteten Schülerleistungen handelt es sich um Lösungsskizzen. Oft sind aber Lösungsvarianten möglich, die in der Skizze nur zum Teil beschrieben werden konnten. Grundsätzlich gilt deshalb, dass alle Varianten, die zu richtigen Lösungen führen, mit voller Punktzahl bewertet werden, unabhängig davon, ob die gewählte Variante in der Lösungsskizze aufgeführt ist oder nicht.

Kursiv gedruckte Passagen sind Hinweise an die korrigierenden Lehrkräfte. Sie sind nicht Bestandteile der erwarteten Schülerleistung.

Aufgabe I: Schiffskarten (Python-Version)

Schwerpunkt: Objektorientierte Programmierung und Modellierung

	Lösungsskizze	Zuordnung Bewertung		
		I	II	III
a)	Es müssen Objekte der Klassen `Schiff`, `Hafen` und `AISStation` erzeugt werden.		4	
	In der Objektorientierten Programmierung ist eine Klasse die grundlegende Struktur für die Definition von Objekten. Sie dient als Bauplan, um Objekte zu erstellen, die bestimmte Eigenschaften (Attribute) und Verhaltensweisen (Methoden) haben. In diesem Projekt werden beispielsweise Objekte für konkrete Schiffe erzeugt, die gleiche Attribute und Methoden haben, aber unterschiedliche Attributwerte wie Position und Name.	3	3	
b)	Es gibt in dem Modell Nutzungsbeziehungen (*hat-ein-Beziehung*) zwischen Elementverwaltung und den Elementen und von den Elementen zur Zeichenfläche. Zudem werden einige Klassen von Oberklassen abgeleitet (*Vererbung, Generalisierung/Spezialisierung, ist-ein-Beziehung*): Ein allgemeiner Typ `Element` für alles, was auf der Karte angezeigt wird, ist Grundlage für die Klassen `Schiff`, `Hafen` und `AISStation`.	2	4	
	Die Vererbungsbeziehung erkennt man im Kopf der Klassendefinition an der Angabe der Oberklasse in Klammern hinter dem Klassennamen, konkret also beispielsweise `class Schiff(Element):` Die Verwendungsbeziehung erkennt man im Programmtext an einem Attribut der verwendenden Klasse oder einer Zugriffsmethode auf ein Objekt der verwendeten Klasse.	4		
	In einer abstrakten Klasse werden die gemeinsamen Eigenschaften anderer Klassen ausmodelliert, ohne dass es Instanzen dieser Klasse geben kann bzw. soll. Man kann also eine Oberklasse implementieren und verhindern, dass ein Objekt von einem allgemeinen Typ `Element` erzeugt wird, für das es bspw. keine Form der Darstellung geben kann. In der abstrakten Klasse kann aber erzwungen werden, dass in allen Unterklassen eine Methode implementiert ist, die für die Darstellung sorgt.	3	3	
c)	In einer Unterklasse kann für den konkreten Schiffstyp die Farbe der Darstellung einheitlich festgelegt werden, indem sie im Konstruktor gesetzt wird. Ebenso kann aber in einer allgemeinen Klasse `Schiff` die Farbe vom Wert des Attributs `schiffstyp` abgeleitet werden.			4
	Nach dem Anlegen müsste das Objekt, das ein Schiff repräsentiert, entfernt werden und ein neues Objekt mit identischen Objektwerten, aber von einem anderen Typ erzeugt werden. Ebenso müsste umgekehrt verfahren werden, wenn ein Schiff wieder ausläuft. Das widerspricht der Idee der objektorientierten Modellierung, nach der ein Objekt über die Lebensdauer seinen Zustand, nicht aber seinen Typ ändert.		3	2

d)	Grundsätzlich stellt dieses Vorgehen eine mögliche Lösung der Anforderung dar, alle Schiffe in einer Datenstruktur zu halten. Es ist aber nicht sinnvoll, die Datenstruktur Tupel zu nutzen, da die Zahl der Schiffe auf der Karte variabel ist		2	2
e)	Die Methode erhält als Parameter die MMSI-Nummer des Schiffs, das entfernt werden soll. Mittels einer `for-Schleife` wird jedes Element der `schiffsliste` einmal geprüft: In Zeile 2 wird eine lokale Variable `akt` mit dem jeweils aktuellen Schiff belegt. Dessen MMSI wird mit der übergebenen verglichen. Ist die MMSI identisch, wird das Schiff aus der Liste `schiffsliste` entfernt (Z. 4) und die Bearbeitung abgebrochen (Z. 5). Im anderen Fall wird die Prüfung mit dem nächsten Schiff fortgesetzt, bis das Ende der Liste erreicht wird: Die Schleife wird ausgeführt, solange die Laufvariable `i` kleiner ist als die Länge der Liste. *Je nach unterrichtlichen Voraussetzungen kann der Abbruch der Schleife mit `return` übergangen werden.*		5	
	Die Verwendung der einmaligen MMSI-Nummer sorgt dafür, dass das richtige Schiff eindeutig erkannt wird.			2
f)	Notwendig sind Attribute für die Länge und die Breite des Schiffs insgesamt. Zudem muss ein Punkt als Ursprung der Form definiert werden, von dem aus die Position der GPS-Antenne gerechnet wird. *Das kann bspw. die Spitze am Bug sein oder die Backbord-Ecke des Hecks.* Hierfür sind zwei weitere Attribute für die Abstände in den beiden Dimensionen notwendig.		2	2
	Insgesamt 50 BE	12	26	12

Aufgabe I: Schiffskarten (Java-Version)

Schwerpunkt: Objektorientierte Programmierung und Modellierung

	Lösungsskizze	Zuordnung Bewertung		
		I	II	III
a)	Es müssen Objekte der Klassen `Schiff`, `Hafen` und `AISStation` erzeugt werden.		4	
	In der Objektorientierten Programmierung ist eine Klasse die grundlegende Struktur für die Definition von Objekten. Sie dient als Bauplan, um Objekte zu erstellen, die bestimmte Eigenschaften (Attribute) und Verhaltensweisen (Methoden) haben. In diesem Projekt werden beispielsweise Objekte für konkrete Schiffe erzeugt, die gleiche Attribute und Methoden haben, aber unterschiedliche Attributwerte wie Position und Name.	3	3	
b)	Es gibt in dem Modell Nutzungsbeziehungen (*hat-ein-Beziehung*) zwischen Elementverwaltung und den Elementen und von den Elementen zur Zeichenfläche. Zudem werden einige Klassen von Oberklassen abgeleitet (*Vererbung, Generalisierung/Spezialisierung, ist-ein-Beziehung*): Ein allgemeiner Typ `Element` für alles, was auf der Karte angezeigt wird, ist Grundlage für die Klassen `Schiff`, `Hafen` und `AISStation`.	2	4	
	Die Vererbungsbeziehung erkennt man im Kopf der Klassendefinition an der Angabe `extends <Oberklasse>`, konkret also beispielsweise `public class Schiff extends Element;` Die Verwendungsbeziehung erkennt man im Programmtext an einem Attribut der verwendenden Klasse oder einer Zugriffsmethode auf ein Objekt der verwendeten Klasse.	4		
	In einer abstrakten Klasse werden die gemeinsamen Eigenschaften anderer Klassen ausmodelliert, ohne dass es Instanzen dieser Klasse geben kann bzw. soll. Man kann also eine Oberklasse implementieren und verhindern, dass ein Objekt von einem allgemeinen Typ `Element` erzeugt wird, für das es bspw. keine Form der Darstellung geben kann. In der abstrakten Klasse kann aber erzwungen werden, dass in allen Unterklassen eine Methode implementiert ist, die für die Darstellung sorgt.	3	3	
c)	In einer Unterklasse kann für den konkreten Schiffstyp die Farbe der Darstellung einheitlich festgelegt werden, indem sie im Konstruktor gesetzt wird. Ebenso kann aber in einer allgemeinen Klasse `Schiff` die Farbe vom Wert des Attributs `schiffstyp` abgeleitet werden.			4
	Nach dem Anlegen müsste das Objekt, das ein Schiff repräsentiert, entfernt werden und ein neues Objekt mit identischen Objektwerten, aber von einem anderen Typ erzeugt werden. Ebenso müsste umgekehrt verfahren werden, wenn ein Schiff wieder ausläuft. Das widerspricht der Idee der objektorientierten Modellierung, nach der ein Objekt über die Lebensdauer seinen Zustand, nicht aber seinen Typ ändert.		3	2
d)	Grundsätzlich stellt dieses Vorgehen eine mögliche Lösung der Anforderung dar, alle Schiffe in einer Datenstruktur zu halten. Es ist		2	2

	aber nicht sinnvoll, die Datenstruktur Array zu nutzen, da die Zahl der Schiffe auf der Karte variabel ist.			
e)	Die Methode erhält als Parameter die MMSI-Nummer des Schiffs, das entfernt werden soll. In Zeile 3 wird eine lokale Variable `akt` vom Typ Schiff angelegt. In der folgenden for-Schleife wird die Schiffsliste vom Anfang an durchgegangen. Das jeweils betrachtete Schiff wird in der Variablen `akt` gehalten und dessen MMSI mit der übergebenen verglichen. Ist die MMSI identisch, wird das Schiff aus der ArrayList `schiffsliste` entfernt (Z. 10) und die Bearbeitung abgebrochen (Z. 11). Im anderen Fall wird die Prüfung mit dem nächsten Schiff fortgesetzt, bis das Ende der ArrayList erreicht wird: Die Schleife wird ausgeführt, solange `i < schiffsliste.size()` gilt. *Je nach unterrichtlichen Voraussetzungen kann der Abbruch der Schleife mit `return;` übergangen werden.*		5	
	Die Verwendung der einmaligen MMSI-Nummer sorgt dafür, dass das richtige Schiff eindeutig erkannt wird.			2
f)	Notwendig sind Attribute für die Länge und die Breite des Schiffs insgesamt. Zudem muss ein Punkt als Ursprung der Form definiert werden, von dem aus die Position der GPS-Antenne gerechnet wird. *Das kann bspw. die Spitze am Bug sein oder die Backbord-Ecke des Hecks.* Hierfür sind zwei weitere Attribute für die Abstände in den beiden Dimensionen notwendig.		2	2
	Insgesamt 50 BE	12	26	12

Aufgabe II: Bitcoin (Python-Version)

Schwerpunkt: Datensicherheit in verteilten Systemen

	Lösungsskizze	Zuordnung Bewertung		
		I	II	III
a)	Es wird dadurch ausschließlich die Integrität der Daten bestätigt.	6		
	Der Hash ist öffentlich einsehbar, genauso wie die Transaktionsdaten. Damit ist Vertraulichkeit und Authentizität nicht gefragt. Wichtig ist lediglich die Integrität der Daten, die auch im Nachhinein sichergestellt sein muss. Da der Hash eines Vorgänger-Blocks jeweils in den nachfolgenden Block mit eingeht, ist es nachträglich nicht möglich, einen vorhergehenden Block zu ändern, ohne dass es in nachfolgenden Blöcken auffallen würde. *Anmerkung: Mit dem Block-Hash wird lediglich der Header eines Blockes verifiziert, die einzelnen Transaktionen in dem Block, also die Transaktionsdaten, werden nochmal separat gehasht und der Hash dann in den Blockhash integriert – diese Unterscheidung wird jedoch von den SuS nicht erwartet, um diese Aufgabe korrekt zu beantworten.*	2	4	
b)	Durch die (modulo … 16777216) Rechnung wird jeder resultierende Hash auf die max. Zahl 16777215 begrenzt. Damit ist die max. Länge des Hashes vorgegeben.	3	4	
	Die `einfacherHash`-Funktion erhält als Eingabe einen Text (`daten`) und eine Zahl (`faktor`). Ziel ist es, aus dem Text eine Zahl zu berechnen, die eine maximale Größe nicht überschreitet, egal wie lang der ursprüngliche Text war. Die Funktion arbeitet mit drei Variablen: einem `index` (startet bei 0), einem Multiplikator `multi` (startet bei 1) und einer Summe `summe` (startet bei 0). Sie durchläuft den String `daten` Zeichen für Zeichen. In jedem Schritt wird der neue Wert von `summe` berechnet nach der Formel: `(summe + (charValue * (multi + index))) % MOD`. Dabei bezeichnet `summe` in dem Klammerausdruck den jeweils vorherigen Wert, und `index` wird nach jedem Durchlauf um 1 erhöht. Damit die Zahlen nicht zu groß werden, besonders bei langen Texten, wird eine Modulo-Operation verwendet. Die Konstante `MOD` ist auf 16777216 gesetzt. Sowohl der `multi`-Wert als auch die Zwischenergebnisse werden durch diese Modulo-Rechnung begrenzt, sodass sie nie größer als `MOD` werden können. Wenn alle Zeichen aus `daten` abgearbeitet sind, wird `summe` zurück an den Aufrufer gegeben. So entsteht aus jedem beliebigen Text eine Zahl fester Maximalgröße - das ist das Grundprinzip einer Hash-Funktion.	1	4	4

c)	Ähnliche Daten sollten keine ähnlichen Hashes erzeugen, weil man sich dem Ergebnis sonst Schritt für Schritt nähern könnte. So aber ist es ein zufälliger Prozess und man muss wie beim Brute-Force Angriff alle Möglichkeiten durchprobieren, bis man zufällig erfolgreich ist.		3	4
d)	Die Funktion `mineBlock` berechnet einen Hash in Hexadezimal-Darstellung aus den gegebenen Daten `daten`, dem vorhergehenden Hash `prevHash` , den Parametern `schwierigkeit`, `maxStellenHash` und `factor`. Zunächst wird mit einigen dieser Parameter mithilfe der Funktion `einfacherHash` ein erster Hashwert der Daten berechnet. In der While-Schleife wird mit der Funktion `hashTrifftSchwierigkeit` geprüft, ob dieser Hashwert bereits die Anforderungen erfüllt, sonst wird der Wert `nonce` in jedem Schritt um 1 hochgezählt und ein neuer String mit `daten + prevHash + str(nonce)` gebildet. Damit wird jedes Mal ein anderer Input für `einfacherHash` erzeugt, weil `nonce` variiert. Die While-Schleife erzeugt solange neue Hashwerte, bis die gewünschte Schwierigkeit erfüllt ist. Hierzu prüft die Funktion `hashTrifftSchwierigkeit` bei jedem Schleifendurchgang, ob bereits ein berechneter Hash den geforderten Schwierigkeitsgrad erfüllt. Hierzu übernimmt sie den Parameter `schwierigkeit`, die angibt, wieviel führende Nullen der hexadezimale Hash am Ende der Berechnung haben muss. Die führenden Nullen entstehen bei der Umrechnung des dezimalen Hashes in die Hexadezimal-Darstellung, da die Funktion `intToHexString` den hexadezimalen String nach der Umrechnung über die Funktion `format` vorne mit Nullen auffüllt, wenn der errechnete hexString kürzer ist als die maximale Länge, die in mineBlock über den Parameter `maxStellenHash` angegeben wird. Sobald die `hashTrifftSchwierigkeit` Funktion `True` liefert, ist die While-Schleife beendet und die Funktion `mineBlock` returniert eine Liste mit dem `nonce` Wert, dem errechneten finalen `hashWert` und dem dazu passenden hexadezimalen Hash.		7	3
e)	„Proof-of-Work-Prozesses" ist eine sinnvolle Bezeichnung für den Prozess des Minens, weil man einen passenden Hash nicht über einen bestimmten Algorithmus finden kann, sondern ihn nur über Ausprobieren und Ausdauer mit verschiedenen Nonces findet – dieser Vorgang verbraucht Rechenzeit und ist daher arbeitsintensiv (für den Rechner).		3	2
	Insgesamt 50 BE	12	25	13

Aufgabe II: Bitcoin (Java-Version)

Schwerpunkt: Datensicherheit in verteilten Systemen

	Lösungsskizze	Zuordnung Bewertung		
		I	II	III
a)	Es wird dadurch ausschließlich die Integrität der Daten bestätigt.	6		
	Der Hash ist öffentlich einsehbar, genauso wie die Transaktionsdaten. Damit ist Vertraulichkeit und Authentizität nicht gefragt. Wichtig ist lediglich die Integrität der Daten, die auch im Nachhinein sichergestellt sein muss. Da der Hash eines Vorgänger-Blocks jeweils in den nachfolgenden Block mit eingeht, ist es nachträglich nicht möglich, einen vorhergehenden Block zu ändern, ohne dass es in nachfolgenden Blöcken auffallen würde *Anmerkung: Mit dem Block-Hash wird lediglich der Header eines Blockes verifiziert, die einzelnen Transaktionen in dem Block, also die Transaktionsdaten, werden nochmal separat gehasht und der Hash dann in den Blockhash integriert – diese Unterscheidung wird jedoch von den SuS nicht erwartet, um diese Aufgabe korrekt zu beantworten.*	2	4	
b)	Durch die (modulo … 16777216) Rechnung wird jeder resultierende Hash auf die max. Zahl 16777215 begrenzt. Damit ist die max. Länge des Hashes vorgegeben.	3	4	
	Die `einfacherHash`-Funktion erhält als Eingabe einen Text (`daten`) und eine Zahl (`faktor`). Ziel ist es, aus dem Text eine Zahl zu berechnen, die eine maximale Größe nicht überschreitet, egal wie lang der ursprüngliche Text war. Die Funktion arbeitet mit drei Variablen: einem `index` (startet bei 0), einem Multiplikator `multi` (startet bei 1) und einer Summe `summe` (startet bei 0). Sie durchläuft den String `daten` Zeichen für Zeichen. In jedem Schritt wird der neue Wert von `summe` berechnet nach der Formel: `(summe + (charValue * (multi + index))) % MOD`. Dabei bezeichnet `summe` in dem Klammerausdruck den jeweils vorherigen Wert, und `index` wird nach jedem Durchlauf um 1 erhöht. Damit die Zahlen nicht zu groß werden, besonders bei langen Texten, wird eine Modulo-Operation verwendet. Die Konstante `MOD` ist auf 16777216 gesetzt. Sowohl der `multi`-Wert als auch die Zwischenergebnisse werden durch diese Modulo-Rechnung begrenzt, sodass sie nie größer als `MOD` werden können. Wenn alle Zeichen aus `daten` abgearbeitet sind, wird `summe` zurück an den Aufrufer gegeben. So entsteht aus jedem beliebigen Text eine Zahl fester Maximalgröße - das ist das Grundprinzip einer Hash-Funktion.	1	4	4

c)	Ähnliche Daten sollten keine ähnlichen Hashes erzeugen, weil man sich dem Ergebnis sonst Schritt für Schritt nähern könnte. So aber ist es ein zufälliger Prozess und man muss wie beim Brute-Force alle Möglichkeiten durchprobieren, bis man zufällig erfolgreich ist.		3	4
d)	Die Funktion `mineBlock` berechnet einen Hash in Hexadezimal-Darstellung aus den gegebenen Daten `daten`, dem vorhergehenden Hash `prevHash` , den Parametern `schwierigkeit`, `maxStellenHash` und `factor`. Zunächst wird mit einigen dieser Parameter mithilfe der Funktion `einfacherHash` ein erster Hashwert der Daten berechnet. In der While-Schleife wird mit der Funktion `hashTrifftSchwierigkeit` geprüft, ob dieser Hashwert bereits die Anforderungen erfüllt, sonst wird der Wert `nonce` in jedem Schritt um 1 hochgezählt und ein neuer String mit `daten + prevHash + str(nonce)` gebildet. Damit wird jedes Mal ein anderer Input für `einfacherHash` erzeugt, weil `nonce` variiert. Die While-Schleife erzeugt solange neue Hashwerte, bis die gewünschte Schwierigkeit erfüllt ist. Hierzu prüft die Funktion `hashTrifftSchwierigkeit` bei jedem Schleifendurchgang, ob bereits ein berechneter Hash den geforderten Schwierigkeitsgrad erfüllt. Hierzu übernimmt sie den Parameter `schwierigkeit`, die angibt, wieviel führende Nullen der hexadezimale Hash am Ende der Berechnung haben muss. Die führenden Nullen entstehen bei der Umrechnung des dezimalen Hashes in die Hexadezimal-Darstellung, da die Funktion `intToHexString` den hexadezimalen String nach der Umrechnung über die Funktion `format` vorne mit 0en auffüllt, wenn der errechnete hexString kürzer ist als die maximale Länge, die in mineBlock über den Parameter `maxStellenHash` angegeben wird. Sobald die `hashTrifftSchwierigkeit` Funktion `true` liefert, ist die While-Schleife beendet und die Funktion `mineBlock` returniert eine Liste mit dem `nonce` Wert und dem errechneten finalen `hashWert`. (*Die hexadezimale Darstellung von `hashWert` wird in `hexHash` festgehalten, dieser lässt sich aber nicht zusätzlich in dem `long`-Array returnieren, weil die Datentypen (`String` und `long`) nicht kompatibel sind.*)		7	3
e)	„Proof-of-Work-Prozesses" ist eine sinnvolle Bezeichnung für den Prozess des Minens, weil man einen passenden Hash nicht über einen bestimmten Algorithmus finden kann, sondern ihn nur über Ausprobieren und Ausdauer mit verschiedenen Nonces findet – dieser Vorgang verbraucht Rechenzeit und ist daher arbeitsintensiv (für den Rechner).		3	2
	Insgesamt 50 BE	12	25	13

Aufgabe III: MENACE (Python-Version)

Schwerpunkt: Intelligente Suchverfahren

	Lösungsskizze	Zuordnung Bewertung		
		I	II	III
a)	Im ersten Schritt werden alle Spielstände erzeugt, die für den ersten Kreis möglich sind. Das sind neun Möglichkeiten. Zu jedem dieser Zustände werden danach die acht möglichen Positionen mit dem ersten Kreuz auf einem der freien Felder bestimmt, insgesamt also 72 Varianten. Zu denen werden jeweils die sieben freien Felder mit dem Kreis belegt. Dadurch entstehen 504 Varianten. Hier sind zwei Aspekte zu berücksichtigen: Einmal unterscheiden sich nicht die beiden Spielsituationen, bei denen die beiden Kreise zwar in unterschiedlicher Reihenfolge, allerdings auf dieselben Felder gesetzt worden sind. Weiterhin sollte erkannt werden, dass die Drehung des Felds um 90°, 180° und 270° genau so wenig zu einer echten neuen Spielsituation führt wie alle Spiegelungen, auch die an den Diagonalen.	3	3 2	4
b)	Die SuS sollen beschreiben, dass mit einer tiefen Liste gearbeitet wird (*sie brauchen diesen Fachausdruck nicht zu nennen, die Schachtelung soll aber im Text erkennbar sein*). Sie sollen beschreiben, dass am Muster erkennbar ist, dass die Teillisten für die Zeilen der Darstellung stehen. Die SuS können angeben, dass auch eine Version mit Tupeln möglich ist, müssen aber erläutern, dass Listen besser geeignet sind, da sie eine veränderbare Datenstruktur darstellen. Für die Tupel müssen sie darauf hinweisen, dass diese statisch sind und deshalb der Spielverlauf nur schlecht behandelt werden kann.	4	2 2	2
c)	Die Funktion durchläuft iterativ die Zeilen und prüft zunächst das erste Zeichen, ob es vom Leerzeichen verschieden ist, und dann die jeweils nachfolgenden Zeichen, ob sie gleich dem ersten sind. Ist das der Fall, gibt sie `True` zurück, anderenfalls `False`. Die Schülerinnen und Schüler sollen erkennen, dass es mit Hilfe einer passenden Hilfsfunktion `holeSpalte(spaltenIndex)` möglich wird, die Prüfung genauso wie bei der Prüfung der Zeile umzusetzen.		3 2	2

d)	Die Funktion füllt das Feld abwechselnd mit den beiden Zeichen 'o' und 'x'.
	Angabe der Rückgabe:
	`spielstand=[`
	`  ['o','x','o'],`
	`  ['x','o','x'],`
	`  ['o','x','o']`
	`]`
	Für eine Tiefensuche fehlt das Generieren und Untersuchen von Alternativen in jedem Schritt.
		3	2
			2
				1
				1

e)	Es werden alle noch freien Felder des Spielstands bestimmt, um dann eine davon für die Fortsetzung des Setzens auswählen zu können. Dabei ist notwendig zu prüfen, ob es überhaupt noch freie Felder gibt, da es sonst einen unzulässigen Zugriff bei `waehleEineZufaellig(positionen)` geben könnte, andererseits auch das Programm beispielsweise die Information verarbeiten muss, dass kein Spieler gewonnen hat.	3	2	
	Bei einer realen Spielsituation wählen die Spieler in der Regel nicht zufällig, sondern nach eigenen Kriterien. Dadurch wird es in der Regel zu anderen bevorzugten Auswahlen und damit zu anderen Zügen kommen.		3	
	Auch durch das zufällige Auswählen kann das Programm eine sinnvolle Bewertung der auftretenden Spielstände ausführen, was die Möglichkeit bietet, eher erfolgreiche weitere Schritte von schlechteren zu unterscheiden.		2	2
	Insgesamt 50 BE	13	25	12

Aufgabe III: MENACE (Java-Version)

Schwerpunkt: Intelligente Suchverfahren

	Lösungsskizze	Zuordnung Bewertung		
		I	II	III
a)	Im ersten Schritt werden alle Spielstände erzeugt, die für den ersten Kreis möglich sind. Das sind neun Möglichkeiten. Zu jedem dieser Zustände werden danach die acht möglichen Positionen mit dem ersten Kreuz auf einem der freien Felder bestimmt, insgesamt also 72 Varianten. Zu denen werden jeweils die sieben freien Felder mit dem Kreis belegt. Dadurch entstehen 504 Varianten. Hier sind zwei Aspekte zu berücksichtigen: Einmal unterscheiden sich nicht die beiden Spielsituationen, bei denen die beiden Kreise zwar in unterschiedlicher Reihenfolge, allerdings auf die selben Felder gesetzt worden sind. Weiterhin sollte erkannt werden, dass die Drehung des Felds um 90°, 180° und 270° genau so wenig zu einer echten neuen Spielsituation führt wie alle Spiegelungen, auch die an den Diagonalen.	3	3 2	4
b)	Die SuS sollen angeben, dass es sich um ein zweidimensionales (tiefes) Array handelt. (*sie brauchen diesen Fachausdruck nicht zu nennen, die Schachtelung soll aber im Text erkennbar sein*).Die SuS können auf die einfachere Lesbarkeit der ersten Version eingehen und erläutern, dass auf flache Arrays zwar einfacher zugegriffen werden kann, dass ein zweidimensionales Array aber besser die Problemstruktur beschreibt und dadurch die Art der Zugriffe verständlicher ist.	4	2 2	2
c)	Die Funktion durchläuft iterativ die Zeilen und prüft zunächst das erste Zeichen, ob es vom Leerzeichen verschieden ist und dann die jeweils nachfolgenden Zeichen, ob sie gleich dem ersten sind. Ist das der Fall, gibt sie `true` zurück, anderenfalls `false`. Die Schülerinnen und Schüler sollen erkennen, dass es mit Hilfe einer passenden Hilfsfunktion `char[] holeSpalte(int spaltenIndex)` möglich wird, die Prüfung genau so wie bei der Prüfung der Zeile umzusetzen.		3 2	2

d)	Die Funktion füllt das Feld zeilenweise abwechselnd mit den beiden Zeichen 'o' und 'x'.
	Angabe einer Ausgabe:
	`- - -`
	`|o|x|o|`
	`- - -`
	`|x|o|x|`
	`- - -`
	`|o|x|o|`
	`- - -`
	Für eine Tiefensuche fehlt das Generieren und Untersuchen von Alternativen in jedem Schritt.
		3	2
			2
				1
				1

e)	Es werden alle noch freien Felder des Spielstands bestimmt, um dann eine davon für die Fortsetzung des Setzens auswählen zu können. Dabei ist notwendig zu prüfen, ob es überhaupt noch freie Felder gibt, da es sonst einen unzulässigen Zugriff bei `waehleEineZufaellig(positionen)` geben könnte, andererseits auch das Programm beispielsweise die Information verarbeiten muss, dass kein Spieler gewonnen hat.	3	2	
	Bei einer realen Spielsituation wählen die Spieler in der Regel nicht zufällig, sondern nach eigenen Kriterien. Dadurch wird es in der Regel zu anderen bevorzugten Auswahlen und damit zu anderen Zügen kommen.		3	
	Auch durch das zufällige Auswählen kann das Programm eine sinnvolle Bewertung der auftretenden Spielstände ausführen, was die Möglichkeit bietet, eher erfolgreiche weitere Schritte von schlechteren zu unterscheiden.		2	2
	Insgesamt 50 BE	13	25	12

Hamburg | Behörde für Schule, Familie und Berufsbildung

_____ / _____
Name / Kurs-Nr.

Schriftliche Abiturprüfung
Schuljahr 2025/2026

Informatik

auf grundlegendem Anforderungsniveau

an allgemeinbildenden und beruflichen gymnasialen Oberstufen

Haupttermin
Mittwoch, 22. April 2026, 09:00 Uhr

Unterlagen für die Prüflinge

Allgemeine Arbeitshinweise

- Überprüfen Sie diese Unterlagen auf Vollständigkeit.
- Schreiben Sie auf alle Prüfungsunterlagen Ihren Namen und zusätzlich auf dieses Deckblatt Ihre Kursnummer.
- Kennzeichnen Sie Ihre Entwurfsblätter (Kladde) und Ihre Reinschrift.

Fachspezifische Arbeitshinweise[1]

- Die Arbeitszeit beträgt **255 Minuten**.
- Eine gesonderte Lese- oder Auswahlzeit wird nicht gewährt.
- Hilfsmittel: Taschenrechner (nicht programmierbar, nicht grafikfähig), Rechtschreibwörterbuch

Aufgabenauswahl

- Sie erhalten **drei** Aufgaben zu unterschiedlichen Schwerpunkten.
- Bearbeiten Sie die **Pflichtaufgabe** und wählen Sie **eine** weitere Aufgabe zur Bearbeitung aus.
- Vermerken Sie auch auf Ihrer Reinschrift, welche Aufgaben Sie ausgewählt und bearbeitet haben.

Bearbeitet wurden die folgenden Aufgaben (Bitte kreuzen Sie an):

I	Objektorientierte Modellierung und Programmierung (Pflicht)	(☐ Python ☐ Java)
II	Datensicherheit in verteilten Systemen	(☐ Python ☐ Java)
III	Intelligente Suchverfahren	(☐ Python ☐ Java)

[1] Entsprechend der „Richtlinie über die Gewährung von Erleichterungen für neu zugewanderte Schülerinnen, Schüler und Prüflinge bei Sprachschwierigkeiten in der deutschen Sprache“ (MBlSchul Nr. 08, 7. Oktober 2016, S. 60) werden für die betroffenen Prüflinge die folgenden Erleichterungen gewährt:
- Die Bearbeitungszeit wird um 30 Minuten auf **285 Minuten** erhöht.
- Ein nicht-elektronisches Wörterbuch Deutsch – Herkunftssprache / Herkunftssprache – Deutsch wird bereitgestellt.

Bewertung

Jeder Aufgabe sind 50 Bewertungseinheiten (BE) zugeordnet. In allen Teilaufgaben werden nur ganze oder halbe BE vergeben. Insgesamt sind 100 BE erreichbar. Bei der Festlegung von Notenpunkten gilt die folgende Tabelle.

Erbrachte Leistung (in BE)	Notenpunkte
≥ 95	15
≥ 90	14
≥ 85	13
≥ 80	12
≥ 75	11
≥ 70	10
≥ 65	9
≥ 60	8

Erbrachte Leistung (in BE)	Notenpunkte
≥ 55	7
≥ 50	6
≥ 45	5
≥ 40	4
≥ 33	3
≥ 27	2
≥ 20	1
< 20	0

Für die Erteilung der **Note gut** (11 Punkte) ist mindestens erforderlich, dass mindestens 75 % der erwarteten Gesamtleistung erbracht werden. Dabei muss die Prüfungsleistung in ihrer Gliederung, in der Gedankenführung, in der Anwendung fachmethodischer Verfahren sowie in der fachsprachlichen Artikulation den Anforderungen voll entsprechen.

Für die Erteilung der **Note ausreichend** (5 Punkte) ist mindestens erforderlich, dass mindestens 45 % der erwarteten Gesamtleistung und über den Anforderungsbereich I hinaus Leistungen in einem weiteren Anforderungsbereich erbracht werden.

Die zwei voneinander unabhängigen Aufgaben der Prüfungsaufgabe werden jeweils mit 50 Bewertungseinheiten bewertet. Die erbrachte Gesamtleistung ergibt sich aus der Summe der Bewertungseinheiten in den beiden Aufgaben.

Schwerwiegende und gehäufte Verstöße gegen die sprachliche Richtigkeit oder gegen die äußere Form führen zu einem Abzug von bis zu zwei Punkten in einfacher Wertung. Mangelhafte Gliederung, Fehler in der Fachsprache, Ungenauigkeiten in Zeichnungen oder unzureichende oder falsche Bezüge zwischen Zeichnungen und Text sind als fachliche Fehler zu werten. Ein Abzug für Verstöße gegen die sprachliche Richtigkeit soll nicht erfolgen, wenn diese bereits Gegenstand der fachspezifischen Bewertungsvorgaben sind.

Aufgabe I: Schiffskarten (Python-Version) 50 BE

Schwerpunkt: Objektorientierte Programmierung und Modellierung

Größere Schiffe senden über das AIS[2] in kurzen Zeitabständen ihre aktuelle Position aus, um Kollisionen mit anderen Schiffen zu vermeiden. Apps wie *Vesselfinder* oder *MarineTraffic* stellen diese Positionen auf Karten dar (vgl. Abb. 1). So kann bspw. ein vom Ufer aus entdecktes Schiff identifiziert werden.

Abbildung 1: Darstellung in einer App. Die Kreise stellen unterschiedliche Schiffe dar. Der Anker steht für den Hafen, der Sendemast für die AIS-Funkstation.

In einer neuen App sollen die Schiffe dargestellt werden, die gerade in Hamburg fahren oder im Hafen liegen. Dabei liegt der Fokus zunächst auf dem Hafengebiet, perspektivisch soll auch der Bereich der Elbe von Geesthacht im Osten Hamburgs bis zur Mündung betrachtet werden. In dem dargestellten Gebiet sollen auch die Häfen und die AIS-Empfangsstationen eingezeichnet werden.

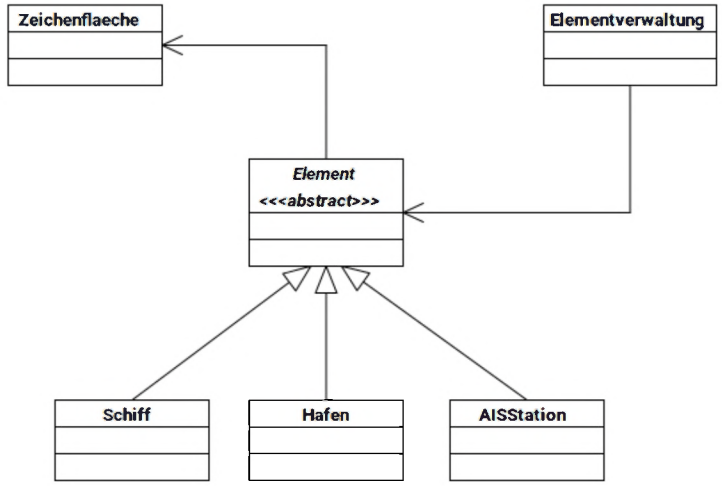


Abbildung 2: Klassendiagramm

I.a • **Geben** Sie **an**, welche Objekte für die Darstellung der `Elemente` auf der Karte in Abb. 1 erzeugt werden müssen.
• **Erläutern** Sie an einem Beispiel aus diesem Projekt, was man unter einer Klasse versteht.

(10 BE)

[2] Das *Automatic Identification System* AIS dient der Sicherheit und Lenkung des Schiffsverkehrs, vgl. Anlage 1.

I.b
- **Erläutern** Sie die Beziehungstypen zwischen den Klassen im Klassendiagramm in Abb. 2.
- **Geben** Sie **an**, woran Sie die unterschiedlichen Beziehungstypen im Programmtext erkennen.
- **Stellen** Sie am Beispiel der Klasse `Element` **dar**, welche Vorteile abstrakte Klassen in der objektorientierten Modellierung haben.

(16 BE)

Unterschiedliche Typen von Schiffen (Handelsschiff, Personenschiff, Tanker, Freizeitboot etc.) sollen auf der Karte in unterschiedlichen Farben dargestellt werden.

Ein fahrendes Schiff soll durch ein Dreieck dargestellt werden. Ein Schiff, das im Hafen oder vor Anker liegt, soll durch einen Kreis symbolisiert werden. Beim fahrenden Schiff soll neben der Position auch die Richtung korrekt in der Karte dargestellt werden. Dafür hat die Klasse Schiff ein Attribut `kurs` vom Typ `int`, in dem der Winkel zwischen Norden und der Fahrtrichtung gespeichert wird. Das darstellende Dreieck wird auf diesen Winkel gedreht und an der richtigen Stelle platziert.

	in Bewegung	gestoppt
Handelsschiff		
Personenschiff		
Tankschiff		
Freizeitboot		

Tabelle 1: Darstellung der verschiedenen Typen von Schiffen in Bewegung und gestoppt.

Zur Umsetzung dieser Anforderung hat die Klasse Schiff ein Attribut `schiffstyp` vom Typ `String` und ein Attribut `inBewegung` vom Typ `boolean`. Alternativ könnten stattdessen Unterklassen für die verschiedenen Schiffstypen von der Klasse `Schiff` abgeleitet werden.

I.c
- **Begründen** Sie, dass beide Varianten eine Lösung für die Anforderung der unterschiedlichen Färbung darstellen.
- **Arbeiten** Sie **heraus**, warum die Unterscheidung fahrender und vor Anker liegender Schiffe nicht über verschiedene Unterklassen geschehen sollte.

(9 BE)

Alle Schiffe, die über die Klasse `Zeichenflaeche` auf dem Bildschirm gezeichnet werden sollen, werden dort in der folgenden Datenstruktur gesammelt:

```
self.schiffsliste = []
```

I.d **Erläutern** Sie, inwieweit die Wahl dieser Datenstruktur sinnvoll ist. Gehen Sie dabei auf die Datenstruktur Tupel ein.

(4 BE)

Im Hamburger Hafen ist stets nur eine kleine Auswahl aller Schiffe vor Ort, die insgesamt auf der Welt unterwegs sind. Trotzdem kann es vorkommen, dass zeitgleich zwei Schiffe mit demselben Namen hier liegen. So lagen am 20.03.25 zwei Schiffe mit dem Namen *Hamburg Express* nur etwa 2 km voneinander entfernt – der Containerfrachter und ein Segelschiff.

In der Klasse Elementverwaltung werden die aktuell registrierten Schiffe in einer Liste gehalten:

```
self.schiffsliste = []
```

I.e
- **Analysieren** Sie die Methode in Anlage 3, die ein aus dem von der App abgedeckten geographischen Bereich herausfahrendes Schiff aus der `schiffsliste` entfernt.
- **Erläutern** Sie, wie in der Implementierung erreicht wird, dass das richtige Schiff aus der `schiffsliste` entfernt wird.

(7 BE)

Die Position eines Schiffs wird mit Hilfe von Satelliten, z. B. des GPS-Systems[3], ermittelt. Die dafür genutzte Antenne ist an einem bestimmten Punkt an Bord eingebaut. Gerade Containerschiffe sind oft so groß, dass diese Position einen Einfluss auf die Positionierung auf der Karte hat und entscheidend dafür sein kann, ob zwei Schiffe aneinander vorbeifahren oder kollidieren. Die *Hamburg Express* ist mit 399 m Länge bspw. so lang, dass der Michel, die größte Kirche in Hamburg, dreimal auf das Schiff passen würde.

In der App soll daher neben den Kreisen und den Dreiecken bei entsprechender Zoomstufe auch der grobe Umriss großer Schiffe dargestellt werden.

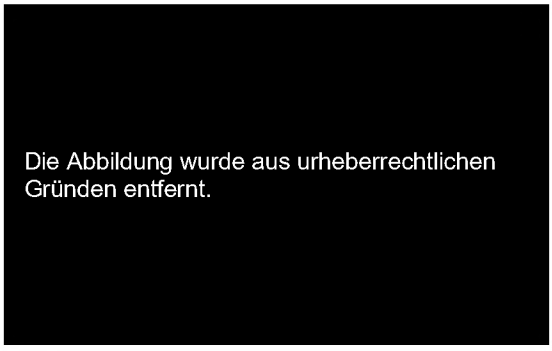


Abbildung 3: Kartenausschnitt mit Schiffen und eingezeichneter Ausdehnung um die gemeldete Position herum

I.f
- **Analysieren** Sie, um welche Attribute Sie die Klasse `Schiff` erweitern würden, um die Größe des Schiffs und die Lage des Schiffs um die ausgesendete Position herum darzustellen.

(4 BE)

[3] *Global Positioning System*, s. Anlage 2

Anlage zur Aufgabe Schiffskarten (Python-Version)

Anlage 1 AIS

Das *Automatic Identification System* AIS dient der Sicherheit und Lenkung des Schiffsverkehrs. Alle Schiffe senden ihre Position, ihren Kurs[4] und ihre Geschwindigkeit sowie ihren Namen, ihre Größe und die MMSI-Nummer[5] – eine weltweit eindeutige 9-stellige Nummer. Bei größeren Schiffen werden auch Angaben über die Ladung und den Zielort ausgesendet. An Bord werden die Aussendungen von anderen Schiffen in der Nähe empfangen. Diese Information wird zur Kollisionsvermeidung genutzt. AIS-Basisstationen an Land erfassen mit Hilfe der Meldungen der Schiffe den Schiffsverkehr im von ihnen abgedeckten Gebiet. Die Reichweite des Signals zwischen zwei Schiffen beträgt ca. 37 km. Entlang der Elbe zwischen Hamburg und der Mündung gibt es neun solcher Stationen, davon drei im Hamburger Stadtgebiet, davon eine in Neumühlen.

In der Karte wird hierfür die folgende Darstellung verwendet: 

Anlage 2 GPS

Das *Global Positioning System* erlaubt es, auf der ganzen Erde jederzeit wetterunabhängig die eigene Position zu bestimmen. Dafür werden Satellitensignale ausgewertet.

Anlage 3 Implementierung der Methode, die ein bestimmtes Schiff aus der Liste entfernt

```
    def entferne_schiff(self, mmsi):
        for akt in self.schiffsliste:
            if akt.gibMMSI() == mmsi:
                self.schiffsliste.remove(akt)
                return
```

[4] Fachbegriff für die „Fahrtrichtung“ eines Schiffs
[5] *Maritime Mobile Service Identity*: Der Containerfrachter Hamburg Express hat bspw. die MMSI-Nummer 211123000.

Aufgabe I: Schiffskarten (Java-Version) 50 BE

Schwerpunkt: Objektorientierte Programmierung und Modellierung

Größere Schiffe senden über das AIS-System[6] in kurzen Zeitabständen ihre aktuelle Position aus, um Kollisionen mit anderen Schiffen zu vermeiden. Apps wie *Vesselfinder* oder *MarineTraffic* stellen diese Positionen auf Karten dar (vgl. Abb. 1). So kann bspw. ein vom Ufer aus entdecktes Schiff identifiziert werden.

Abbildung 1: Darstellung in einer App. Die Kreise stellen unterschiedliche Schiffe dar. Der Anker steht für den Hafen, der Sendemast für die AIS-Funkstation.

In einer neuen App sollen die Schiffe dargestellt werden, die gerade in Hamburg fahren oder im Hafen liegen. Dabei liegt der Fokus zunächst auf dem Hafengebiet, perspektivisch soll auch der Bereich der Elbe von Geesthacht im Osten Hamburgs bis zur Mündung betrachtet werden. In dem dargestellten Gebiet sollen auch die Häfen und die AIS-Empfangsstationen eingezeichnet werden.

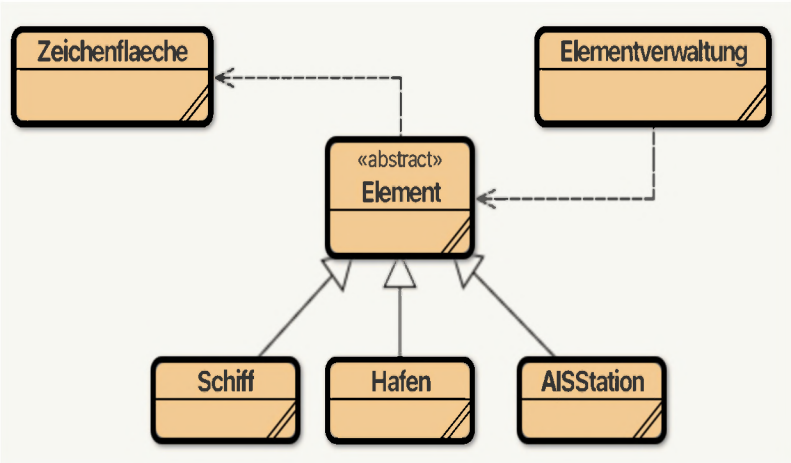


Abbildung 2: Klassendiagramm

I.a
- **Geben** Sie **an**, welche Objekte für die Darstellung der `Elemente` auf der Karte in Abb. 1 erzeugt werden müssen.
- **Erläutern** Sie an einem Beispiel aus diesem Projekt, was man unter einer Klasse versteht.

[6] Das *Automatic Identification System* AIS dient der Sicherheit und Lenkung des Schiffsverkehrs, vgl. Anlage 1.

(10 BE)

I.b
- **Erläutern** Sie die Beziehungstypen zwischen den Klassen im Klassendiagramm in Abb. 2.
- **Geben** Sie **an**, woran Sie die unterschiedlichen Beziehungstypen im Programmtext erkennen.
- **Stellen** Sie am Beispiel der Klasse `Element` **dar**, welche Vorteile abstrakte Klassen in der objektorientierten Modellierung haben.

(16 BE)

Unterschiedliche Typen von Schiffen (Handelsschiff, Personenschiff, Tanker, Freizeitboot etc.) sollen auf der Karte in unterschiedlichen Farben dargestellt werden.

Ein fahrendes Schiff soll durch ein Dreieck dargestellt werden. Ein Schiff, das im Hafen oder vor Anker liegt, soll durch einen Kreis symbolisiert werden. Beim fahrenden Schiff soll neben der Position auch die Richtung korrekt in der Karte dargestellt werden. Dafür hat die Klasse Schiff ein Attribut `kurs` vom Typ `int`, in dem der Winkel zwischen Norden und der Fahrtrichtung gespeichert wird. Das darstellende Dreieck wird auf diesen Winkel gedreht und an der richtigen Stelle platziert.

	in Bewegung	gestoppt
Handelsschiff		
Personenschiff		
Tankschiff		
Freizeitboot		

Tabelle 2: Darstellung der verschiedenen Typen von Schiffen in Bewegung und gestoppt.

Zur Umsetzung dieser Anforderung hat die Klasse Schiff ein Attribut `schiffstyp` vom Typ `String` und ein Attribut `inBewegung` vom Typ `boolean`. Alternativ könnten stattdessen Unterklassen für die verschiedenen Schiffstypen von der Klasse `Schiff` abgeleitet werden.

I.c
- **Begründen** Sie, dass beide Varianten eine Lösung für die Anforderung der unterschiedlichen Färbung darstellen.
- **Arbeiten** Sie **heraus**, warum die Unterscheidung fahrender und vor Anker liegender Schiffe nicht über verschiedene Unterklassen geschehen sollte.

(9 BE)

Alle Schiffe, die über die Klasse `Zeichenflaeche` auf dem Bildschirm gezeichnet werden sollen, werden dort in der folgenden Datenstruktur gesammelt:

```
private Schiff[] schiffe;
```

I.d
- **Erläutern** Sie, inwieweit die Wahl dieser Datenstruktur sinnvoll ist. Gehen Sie dabei auf die Datenstruktur ArrayList ein.

(4 BE)

Im Hamburger Hafen ist stets nur eine kleine Auswahl aller Schiffe vor Ort, die insgesamt auf der Welt unterwegs sind. Trotzdem kann es vorkommen, dass zeitgleich zwei Schiffe mit demselben Namen hier liegen. So lagen am 20.03.25 zwei Schiffe mit dem Namen *Hamburg Express* nur etwa 2 km voneinander entfernt – der Containerfrachter und ein Segelschiff.

In der Klasse Elementverwaltung werden die aktuell registrierten Schiffe, anders als in Aufgabe d, in einer Liste gehalten:

```
private ArrayList<Schiffe> schiffsliste;
```

I.e
- **Analysieren** Sie die Methode in Anlage 3, die ein aus dem von der App abgedeckten geographischen Bereich herausfahrendes Schiff aus der `schiffsliste` entfernt.
- **Erläutern** Sie, wie in der Implementierung erreicht wird, dass das richtige Schiff aus der `schiffsliste` entfernt wird.

(7 BE)

Die Position eines Schiffs wird mit Hilfe von Satelliten, z. B. des GPS-Systems[7], ermittelt. Die dafür genutzte Antenne ist an einem bestimmten Punkt an Bord eingebaut. Gerade Containerschiffe sind oft so groß, dass diese Position einen Einfluss auf die Positionierung auf der Karte hat und entscheidend dafür sein kann, ob zwei Schiffe aneinander vorbeifahren oder kollidieren. Die *Hamburg Express* ist mit 399 m Länge bspw. so lang, dass der Michel, die größte Kirche in Hamburg, dreimal auf das Schiff passen würde.

In der App soll daher neben den Kreisen und den Dreiecken bei entsprechender Zoomstufe auch der grobe Umriss großer Schiffe dargestellt werden.

Abbildung 3: Kartenausschnitt mit Schiffen und eingezeichneter Ausdehnung um die gemeldete Position herum

I.f **Analysieren** Sie, um welche Attribute Sie die Klasse `Schiff` erweitern würden, um die Größe des Schiffs und die Lage des Schiffs um die ausgesendete Position herum darzustellen.

(4 BE)

[7] *Global Positioning System*, s. Anlage 2

Anlage zur Aufgabe Schiffkarten (Java-Version)

Anlage 1 AIS

Das *Automatic Identification System* AIS dient der Sicherheit und Lenkung des Schiffsverkehrs. Alle Schiffe senden ihre Position, ihren Kurs[8] und ihre Geschwindigkeit sowie ihren Namen, ihre Größe und die MMSI-Nummer[9] – eine weltweit eindeutige 9-stellige Nummer. Bei größeren Schiffen werden auch Angaben über die Ladung und den Zielort ausgesendet. An Bord werden die Aussendungen von anderen Schiffen in der Nähe empfangen. Diese Information wird zur Kollisionsvermeidung genutzt. AIS-Basisstationen an Land erfassen mit Hilfe der Meldungen der Schiffe den Schiffsverkehr im von ihnen abgedeckten Gebiet. Die Reichweite des Signals zwischen zwei Schiffen beträgt ca. 37 km. Entlang der Elbe zwischen Hamburg und der Mündung gibt es neun solcher Stationen, davon drei im Hamburger Stadtgebiet, davon eine in Neumühlen.

In der Karte wird hierfür die folgende Darstellung verwendet: 

Anlage 2 GPS

Das *Global Positioning System* erlaubt es, auf der ganzen Erde jederzeit wetterunabhängig die eigene Position zu bestimmen. Dafür werden Satellitensignale ausgewertet.

Anlage 3 Implementierung der Methode, die ein bestimmtes Schiff aus der ArrayList entfernt

```
public void entferneSchiff(int mmsi)
{
   Schiff akt;
   for (int i = 0; i < schiffsliste.size(); i++)
   {
      akt = schiffliste.get(i);
      if (akt.gibMMSI() == mmsi)
      {
         schiffsliste.remove(i);
         return;
       }
    }
 }
```

[8] Fachbegriff für die „Fahrtrichtung“ eines Schiffs
[9] *Maritime Mobile Service Identity*: Der Containerfrachter Hamburg Express hat bspw. die MMSI-Nummer 211123000.

Aufgabe II: Bitcoin (Python-Version) 50 BE

Schwerpunkt: Datensicherheit in verteilten Systemen

Die Bitcoin-Blockchain ist eine einzigartige, fortlaufende Kette von Datenblöcken. Jeder Block enthält etwa 1.000 bis 3.000 Transaktionsdaten (Käufe und Verkäufe von Bitcoin), vergleichbar mit Banküberweisungen. Das Besondere an dieser Struktur ist die Art der Verknüpfung: Die Blöcke sind durch spezielle Hashwerte (kurz Hashes[10]) miteinander verbunden, wobei der Hash jedes Blocks in die Berechnung des Hashes des nachfolgenden Blocks einfließt. Diese Verkettung macht nachträgliche Manipulationen der Transaktionsdaten praktisch unmöglich.

Im Gegensatz zur sonst üblichen einmaligen Berechnung eines Hashs für einen Datenblock muss bei der Bitcoin-Blockchain ein spezieller Hash gefunden werden, der in der hexadezimalen Darstellung eine bestimmte Anzahl führender Nullen aufweist. Diese Nullen kommen in der hexadezimalen Darstellung (als String) dadurch zustande, dass eine feste Länge an Zeichen vorgegeben wird, und wenn die umgerechnete Dezimalzahl weniger Hexadezimal-Stellen aufweist, der hexadezimale String einfach vorne mit Nullen aufgefüllt wird. Diese Anzahl führender Nullen für einen „gültigen“ Hash wird durch den Parameter *difficulty* (Schwierigkeit) vorgegeben. Da es nicht möglich ist, vorherzuberechnen, wie man die Daten im Block ändern muss, um einen bestimmten Hash zu erhalten, kann dieses Problem nur durch Ausprobieren gelöst werden. Je mehr führende Nullen gefordert sind, desto höher ist die Schwierigkeit und desto größer der erforderliche Rechenaufwand.

Der Prozess des „Schürfens" (mining) besteht darin, einem Block eine Zahl („Nonce“ – „**N**umber only used **once**“) hinzuzufügen und diese solange zu verändern, bis ein Hash mit der geforderten Anzahl führender Nullen gefunden wird. Diese rechenintensive Aufgabe ist der Kern des Bitcoin-Mining-Prozesses.

Jeder neue Block dieser Bitcoin-Blockchain enthält unter anderem:

- **Transaktionsdaten**: Daten, wie z. B. „Alice sendet 5 BTC an Bertram“.
- **„Previous“ Block-Hash**: Der Hash des vorherigen Blocks, der die Kette sichert.
- **Nonce**: Eine einfache Zahl, die den Transaktionsdaten am Ende hinzugefügt wird. Sie wird während des Mining-Prozesses hochgezählt, um einen passenden, neuen Hash zu finden.
- **„New“ Block-Hash**: Der eigene Hash des Blocks, der durch das Mining neu berechnet wurde und für den die Miner die „Belohnung“ in Bitcoin bekommen.

II.a Hashing ist ein wesentlicher Bestandteil der Blockchain.
• **Geben** Sie in diesem Kontext **an**, ob durch das Hashing die Authentizität, die Integrität und/oder die Vertraulichkeit der in der Blockchain verwalteten Daten sichergestellt werden.
• **Erläutern** Sie, warum die Daten der Blockchain in Bezug auf diese/-n Punkt/-e „sicher“ sind.

(12 BE)

In Anlage I finden Sie die Funktion `einfacherHash`, die ein vereinfachtes Hashing-Verfahren implementiert. Diese Funktion berechnet einen Hashwert in dezimaler Darstellung. Zum Beispiel ergibt der Aufruf `einfacherHash ("Hallo, dies ist ein Test", 31)` den Wert `6266346`.

II.b • **Beschreiben** Sie, wie dieses einfache Hash-Verfahren zu einem Hash mit begrenzter Länge führt.
• **Analysieren** Sie die Funktionsweise von `einfacherHash`.

(16 BE)

[10] ***Hashwert:*** Ein Hashwert (auch Hash genannt) ist eine eindeutige Zeichenfolge fester Länge, die aus beliebigen Daten berechnet wird wie ein digitaler Fingerabdruck. Die Berechnung des Hashs ist schnell und einfach und die Rückrechnung vom Hash zu den ursprünglichen Daten ist praktisch unmöglich (Einwegfunktion).

Der durch die Blockchain-Vorgaben erzwungene Rechenaufwand, um einen „passenden Hash“ zu finden („schürfen“), schafft einen Wettbewerb zwischen den „Minern", die darum wetteifern, den jeweils nächsten Block mit einem passenden Hash zu finden. Erst wenn dieser gefunden ist, kann der Block zur Bitcoin-Kette hinzugefügt werden, da der Hash des Vorgängerblocks Teil des neuen Hashs ist. Diese Aufgabe erfordert je nach festgelegter Schwierigkeit (bestimmt durch die Anzahl der führenden Nullen) erhebliche Rechenleistung, da eine Lösung nur durch das Ausprobieren verschiedener Nonces (zusätzliche Zahlenwerte) gefunden werden kann.
Die Miner investieren diese Rechenleistung aufgrund der in Aussicht gestellten Belohnung in Form von Bitcoin. Mit höherer Rechenkapazität kann ein Miner seine Chancen verbessern, einen gültigen Hash zuerst zu finden und somit in den Genuss der Belohnung zu kommen.

Beispiel (vereinfacht): Es soll ein Hash mit sechs hexadezimalen Stellen (das entspricht 24 bit) erzeugt werden, wobei als Schwierigkeitsgrad mindestens zwei führende Nullen erforderlich sind. Der Hash wird über das Verfahren in **Anlage 1** berechnet:

Daten: "15.05.2025: Alice an Paul 0.003 BTC“
Previous Hash (Hex): "00c2b0"
Nonce: 0

Beim Ausprobieren verschiedener Nonces kommt man z. B. auf folgende Werte:

Nonce Wert 218	→	Hash (Dezimal):	1323898	→	Hexadezimal: 014337
Nonce Wert 219	→	Hash (Dezimal):	40196	→	Hexadezimal: **009d04**

Die Berechnung ist abgeschlossen, da mit dem Hash **009d04** die vorgegebene Schwierigkeit (mindestens zwei führende Nullen in der hexadezimalen Darstellung) erfüllt ist.

II.c **Erklären** Sie anhand dieses Beispiels, warum es beim Hashing so wichtig ist, dass ähnliche Daten keine ähnlichen Hashes erzeugen.

(7 BE)

II.d **Erläutern** Sie anhand der Funktion `mineBlock` (siehe Anlage I), wie ein neuer Block mit einem gültigen neuen Hash versehen wird, welcher der vorgegebenen Schwierigkeit entspricht. Für diese Aufgabe ist es nicht nötig, im Detail auf die Funktionsweise der Funktion `einfacherHash` einzugehen. Es reicht zu verstehen, dass diese Funktion aus verschiedenen, übergebenen Parametern einen Hash fester Länge berechnet.

(10 BE)

Der gefundene, passende Hash, der am Ende den neuen Block verifiziert, ist das Ergebnis des „Proof-of-Work-Prozesses” (Nachweis-von-Arbeit).

II.e **Begründen** Sie, dass „Proof-of-Work“ ein passender Name für diesen Prozess ist.

(5 BE)

Anlage zur Aufgabe Bitcoin (Python-Version)

Anlage 1 Hashingprojekt

Um einen neuen Block der Blockchain mit dem passenden hexadezimalen Hash (mit entsprechend vielen Nullen vorne) hinzufügen zu können, ist folgender Code vorgegeben:

Anmerkungen:
1. Die Python-Funktion: `format(123, 'x')` macht aus einer Dezimalzahl (hier 123) einen hexadezimalen String: `"7b"`.
2. Die Funktion `ord()` berechnet den ASCII-Wert zum übergebenen Zeichen. Z.B. wird der Großbuchstabe A zu 65: `ord('A')` → `65`.
3. In Python ermittelt der Modulo Operator % den Rest, der nach der ganzzahligen Division von `a` durch `b` verbleibt. Beispiel: `9%5` → `4`, da 9:5=1 Rest 4 ist.

```
def einfacherHash(daten, faktor):
    MOD = 16777216
    index = 0
    summe = 0
    multi = 1

    for char in daten:
        charValue = ord(char)
        summe = (summe + (charValue * (multi + index))) % MOD
        index += 1
        multi = (multi * faktor) % MOD

    return summe

# Test
print(einfacherHash ("Hallo, dies ist ein Test", 31)) # -> 6266346


############################################################
#Haupt Mining Funktion
def mineBlock(daten, prevHash, schwierigkeit, maxStellenHash, faktor):
    nonce = 0
    hashWert = einfacherHash(daten + prevHash + str(nonce), faktor)

    while not hashTrifftSchwierigkeit(hashWert, schwierigkeit,
     maxStellenHash):
        nonce += 1
        hashWert = einfacherHash(daten + prevHash + str(nonce), faktor)

    return [
        nonce,
        hashWert,
        intToHexString(hashWert, maxStellenHash)
    ]
```

```
####################################
#Projekt Mining Hilfsfunktionen
def intToHexString(dezimalzahl, laenge):
    hexString = format(dezimalzahl, 'x')
    while len(hexString) < laenge:
        hexString = '0' + hexString
    return hexString

def hashTrifftSchwierigkeit(hashWert, schwierigkeit, maxStellenHash):
    hexString = intToHexString(hashWert, maxStellenHash)
    return checkPrefix(hexString, schwierigkeit)

def checkPrefix(hexString, schwierigkeit):
    for position in range(schwierigkeit):
        if position >= len(hexString) or hexString[position] != '0':
            return False
    return True

# Test
print(mineBlock("Hallo, dies ist ein Test", "006dAA",2,6, 31))
# -> [471, 17084, '0042bc']
```

Aufgabe II: Bitcoin (Java-Version) 50 BE

Schwerpunkt: Datensicherheit in verteilten Systemen

Die Bitcoin-Blockchain ist eine einzigartige, fortlaufende Kette von Datenblöcken. Jeder Block enthält etwa 1.000 bis 3.000 Transaktionsdaten (Käufe und Verkäufe von Bitcoin), vergleichbar mit Banküberweisungen. Das Besondere an dieser Struktur ist die Art der Verknüpfung: Die Blöcke sind durch spezielle Hashwerte (kurz Hashes[11]) miteinander verbunden, wobei der Hash jedes Blocks in die Berechnung des Hashes des nachfolgenden Blocks einfließt. Diese Verkettung macht nachträgliche Manipulationen der Transaktionsdaten praktisch unmöglich.

Im Gegensatz zur sonst üblichen einmaligen Berechnung eines Hashs für einen Datenblock muss bei der Bitcoin-Blockchain ein spezieller Hash gefunden werden, der in der hexadezimalen Darstellung eine bestimmte Anzahl führender Nullen aufweist. Diese Nullen kommen in der hexadezimalen Darstellung (als String) dadurch zustande, dass eine feste Länge an Zeichen vorgegeben wird und, wenn die umgerechnete Dezimalzahl weniger Hexadezimal-Stellen aufweist, der hexadezimale String einfach vorne mit Nullen aufgefüllt wird. Diese Anzahl führender Nullen für einen „gültigen“ Hash wird durch den Parameter *difficulty* (Schwierigkeit) vorgegeben. Da es nicht möglich ist, vorherzuberechnen, wie man die Daten im Block ändern muss, um einen bestimmten Hash zu erhalten, kann dieses Problem nur durch Ausprobieren gelöst werden. Je mehr führende Nullen gefordert sind, desto höher ist die Schwierigkeit und desto größer der erforderliche Rechenaufwand.

Der Prozess des „Schürfens" (mining) besteht darin, einem Block eine Zahl („Nonce“ – „**N**umber only used **once**“) hinzuzufügen und diese solange zu verändern, bis ein Hash mit der geforderten Anzahl führender Nullen gefunden wird. Diese rechenintensive Aufgabe ist der Kern des Bitcoin-Mining-Prozesses.

Jeder neue Block dieser Bitcoin-Blockchain enthält unter anderem:

- **Transaktionsdaten**: Daten, wie z.B. „Alice sendet 5 BTC an Bertram“.
- **„Previous“ Block-Hash**: Der Hash des vorherigen Blocks, der die Kette sichert.
- **Nonce**: Eine einfache Zahl, die den Transaktionsdaten am Ende hinzugefügt wird. Sie wird während des Mining-Prozesses hochgezählt, um einen passenden, neuen Hash zu finden.
- **„New“ Block-Hash**: Der eigene Hash des Blocks, der durch das Mining neu berechnet wurde und für den die Miner die „Belohnung” in Bitcoin bekommen.

II.a Hashing ist ein wesentlicher Bestandteil der Blockchain.
• **Geben** Sie in diesem Kontext **an**, ob durch das Hashing die Authentizität, die Integrität und/oder die Vertraulichkeit der in der Blockchain verwalteten Daten sichergestellt werden.
• **Erläutern** Sie, warum die Daten der Blockchain in Bezug auf diese/-n Punkt/-e „sicher“ sind.

(12 BE)

In Anlage 1 finden Sie die Methode `einfacherHash`, die ein vereinfachtes Hashing-Verfahren implementiert. Diese Methode berechnet einen Hashwert in dezimaler Darstellung. Zum Beispiel ergibt der Aufruf `einfacherHash("Hallo, dies ist ein Test", 31);` den Wert `6266346`.

II.b • **Beschreiben** Sie, wie dieses einfache Hash-Verfahren zu einem Hash mit begrenzter Länge führt.
• **Analysieren Sie** die Funktionsweise von `einfacherHash`.

(16 BE)

[11] **Hashwert**: Ein Hashwert (auch Hash genannt) ist eine eindeutige Zeichenfolge fester Länge, die aus beliebigen Daten berechnet wird - wie ein digitaler Fingerabdruck. Die Berechnung des Hashs ist schnell und einfach und die Rückrechnung vom Hash zu den ursprünglichen Daten ist praktisch unmöglich (Einwegfunktion).

Der durch die Blockchain-Vorgaben erzwungene Rechenaufwand, um einen „passenden Hash“ zu finden („schürfen“), schafft einen Wettbewerb zwischen den „Minern“, die darum wetteifern, den jeweils nächsten Block mit einem passenden Hash zu finden. Erst wenn dieser gefunden ist, kann der Block zur Bitcoin-Kette hinzugefügt werden, da der Hash des Vorgängerblocks Teil des neuen Hashs ist. Diese Aufgabe erfordert je nach festgelegter Schwierigkeit (bestimmt durch die Anzahl der führenden Nullen) erhebliche Rechenleistung, da eine Lösung nur durch das Ausprobieren verschiedener Nonces (zusätzliche Zahlenwerte) gefunden werden kann.

Die Miner investieren diese Rechenleistung aufgrund der in Aussicht gestellten Belohnung in Form von Bitcoin. Mit höherer Rechenkapazität kann ein Miner seine Chancen verbessern, einen gültigen Hash zuerst zu finden und somit in den Genuss der Belohnung zu kommen.

Beispiel (vereinfacht): Es soll ein Hash mit sechs hexadezimalen Stellen erzeugt werden, wobei als Schwierigkeitsgrad mindestens zwei führende Nullen erforderlich sind. Der Hash wird über das Verfahren in **ANLAGE ZUR AUFGABE** berechnet:

Daten: "15.05.2025: Alice an Paul 0.003 BTC“
Previous Hash (Hex): "00c2b0"
Nonce: 0

Beim Ausprobieren verschiedener Nonces kommt man z.B. auf folgende Werte:

Nonce Wert 218	→	Hash (Dezimal):	1323898	→	Hexadezimal: 014337
Nonce Wert 219	→	Hash (Dezimal):	40196	→	Hexadezimal: **009d04**

Die Berechnung ist abgeschlossen, da mit dem Hash **009d04** die vorgegebene Schwierigkeit (mindestens zwei führende Nullen in der hexadezimalen Darstellung) erfüllt ist.

II.c **Erklären** Sie anhand dieses Beispiels, warum es beim Hashing so wichtig ist, dass ähnliche Daten keine ähnlichen Hashes erzeugen.

(7 BE)

II.d **Erläutern** Sie anhand der Methode `mineBlock` in Anlage 1, wie ein neuer Block mit einem gültigen neuen Hash versehen wird, welcher der vorgegebenen Schwierigkeit entspricht. Für diese Aufgabe ist es nicht nötig, im Detail auf die Funktionsweise der Methode `einfacherHash` einzugehen. Es reicht zu verstehen, dass diese Methode aus verschiedenen, übergebenen Parametern einen Hash fester Länge berechnet.

(10 BE)

Der gefundene, passende Hash, der am Ende den neuen Block verifiziert, ist das Ergebnis des „Proof-of-Work-Prozesses” (Nachweis-von-Arbeit).

II.e **Begründen** Sie, dass Proof-of-work ein passender Name für diesen Prozess ist.

(5 BE)

Anlage zur Aufgabe Bitcoin (Java-Version)

Anlage 1 Hashingprojekt

Um einen neuen Block der Blockchain mit dem passenden hexadezimalen Hash (mit entsprechend vielen Nullen vorne) hinzufügen zu können ist folgender Code vorgegeben:

Anmerkungen:

1. Die Java-Methode: `String.valueOf(123)` wandelt eine Dezimalzahl (hier 123) in einen String "123" um.

2. Ebenso wandelt die Methode `Arrays.toString(new long[]{1,2})` ein Array von long-Zahlen in den String: `"[1, 2]"` um.

3. Die Java-Methode `Long.toHexString(123)` wandelt eine Dezimalzahl (hier 123) in einen hexadezimalen String: "7b" um.

4. Die Methode `charAt(1)` liefert aus dem String "ABC" das Zeichen an der Indexposition 1, nämlich 'B'. D. h. `"ABC".charAt(1)` → 'B'.

```
import java.util.Arrays;
public class BlockChain
{
    public BlockChain(){
        // Tests
        System.out.println(
            einfacherHash("Hallo, dies ist ein Test", 31));  //-> 6266346
        long[] nonce_hashWert =
            mineBlock("Hallo, dies ist ein Test", "006dAA",2, 6,  31);
        System.out.println(Arrays.toString(nonce_hashWert));
        //-> "[471, 17084]"
        System.out.println(intToHexString(17084, 6)); //-> "0042bc"
    }

    public int einfacherHash(String daten, int faktor) {
        final int MOD = 16777216;
        long summe = 0;
        long multi = 1;

        for (int index = 0; index < daten.length(); index++) {
            int charValue = (int) daten.charAt(index);
            summe = (summe + (charValue * (multi + index))) % MOD;
            multi = (multi * faktor) % MOD;
        }
        return (int) summe;
    }
```

```
 //#################################################################
    //Haupt-Mining-Funktion
    public long[] mineBlock(String daten, String prevHash,
      int schwierigkeit, int maxStellenHash, int faktor) {
        int nonce = 0;
        long hashWert = einfacherHash(daten + prevHash +
          String.valueOf(nonce), faktor);
        String hexHash = intToHexString(hashWert, maxStellenHash);
        while(!hashTrifftSchwierigkeit(hashWert, schwierigkeit,
            maxStellenHash)) {
            nonce++;
            hashWert = einfacherHash(daten + prevHash +
                       String.valueOf(nonce), faktor);
            hexHash = intToHexString(hashWert, maxStellenHash);
        }
        return new long[] { nonce, hashWert};
    }
    //###################################
    //Projekt Mining Hilfsfunktionen
    public  String intToHexString(long dezimalzahl, int laenge) {
        String hexString = Long.toHexString(dezimalzahl);
        while (hexString.length() < laenge) {
            hexString = "0" + hexString;
        }
        return hexString;
    }
    public  boolean hashTrifftSchwierigkeit(long hashWert,
      int schwierigkeit, int maxStellenHash) {
        String hexString = intToHexString(hashWert, maxStellenHash);
        return checkPrefix(hexString, schwierigkeit);
    }
    public  boolean checkPrefix(String hexString, int schwierigkeit) {
        for (int pos = 0; pos < schwierigkeit; pos++) {
            if (pos >= hexString.length() || hexString.charAt(pos) != '0'){
                return false;
            }
        }
          return true;
    }
}
```

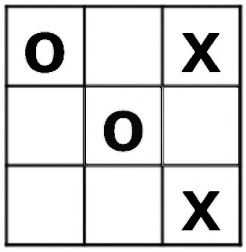
Aufgabe III: MENACE (Python-Version) 50 BE

Schwerpunkt: Intelligente Suchverfahren

Unter dem Namen MENACE hat Donald Michie in den 1960er Jahren eine Idee entwickelt, bei der er, noch ohne einen Computer einzusetzen, das Vorgehen moderner KI-Anwendungen vorwegnahm.

(*Text dazu und zu Tic-Tac-Toe siehe Anlagen 1 und 2*)

Für die in dem Text beschriebene Lernphase sollen in dieser Aufgabe einige Teilschritte modelliert werden.



Teilaufgaben

III.a
- **Beschreiben** Sie, wie man mit Hilfe von Breitensuche alle möglichen Spielzustände nach dem Setzen der ersten drei Zeichen (also zum Beispiel Kreis, Kreuz, Kreis) bestimmen kann.
- **Bestimmen** Sie dabei auch die Anzahl der möglichen verschiedenen Abfolgen beim Setzen der drei Zeichen.
- **Begründen** Sie, dass sich viele dieser Spielzustände prinzipiell (also im Sinn der Spielidee von Tic-Tac-Toe) nicht voneinander unterscheiden.

(12 BE)

III.b
- **Beschreiben** Sie die folgende angegebene Datenstruktur zum Verwalten des jeweiligen Spielstands von Tic-Tac-Toe. Verwenden Sie dazu das oben dargestellte Bild als Beispiel. Dort hat der Spieler 1 (Kreise) bereits zweimal gesetzt und der Spieler 2 (Kreuze) ebenfalls, so dass der Spieler 1 als nächster setzen kann.

```
[['o',' ','x'],[' ','o',' '],[' ',' ','x']]
```

- **Vergleichen** Sie mit einem Einsatz von Tupeln der Form:

```
(('o',' ','x'),(' ','o',' '),(' ',' ','x'))
```

(10 BE)

III.c
- **Erläutern** Sie, wie die im Folgenden angegebene Funktion `hatGewonnenZeile(spielstand)` den aktuellen Spielstand daraufhin überprüft, ob eine der drei Zeilen mit denselben Zeichen belegt ist, so dass der jeweilige Spieler / die jeweilige Spielerin gewonnen hat.

```
def hatGewonnenZeile(spielstand):
    for zeile in spielstand:
        if zeile[0]!=' ':
            if zeile[0]==zeile[1]:
                if zeile[0]==zeile[2]:
                    return True
    return False
```

- **Entwickeln** Sie das Vorgehen für eine Funktion `hatGewonnenSpalte()`.

(7 BE)

Die in der Anlage 3 angegebene Funktion generiert nicht(!) alle Möglichkeiten mit Hilfe von Tiefensuche. Die Funktion startet mit einem leeren Feld.

III.d
- **Analysieren** Sie die Aufgabe der Funktion `generiere(spielstand, zeichen)` in Anlage 3.
- **Geben** Sie **an**, was sie beim Aufruf von `generiere(spielstand, 'o')` generiert.
- **Untersuchen** Sie, was in der Funktion für eine Umsetzung einer Tiefensuche fehlt.

(9 BE)

Das Vorgehen von Donald Michie soll mit dem Computer simuliert werden. Dazu führen wir zu untersuchten Abfolgen von Zügen jeweils eine Bewertung der Spielstände entsprechend dem Vorgehen von Donald Michie ein. Dabei gehen wir so vor, dass in einer ausreichend großen Anzahl von Abfolgen von Zügen schrittweise jede so erzeugt wird, dass bei jedem Spielstand ein noch freies Feld zufällig zum Besetzen mit dem Zeichen des jeweils aktuellen Spielers ausgewählt werden soll. Mit dieser neuen Belegung soll dann jeweils der Zug fortgesetzt werden.[12]

Ein Programm zur Umsetzung dieses Vorgehens enthält die folgenden drei Funktionen:

```
def hatFreiePositionen(spielstand):
    '''Untersucht, ob es im aktuellen Spielstand noch freie Positionen
gibt'''
    . . .

def gibAlleFreienPositionen(spielstand):
    '''gibt (als Paare) die Positionen aller freien Felder als Liste
zurück'''
    . . .

def waehleEineZufaellig(positionen):
    '''wählt eine der Positionen zufällig aus'''
    . . .
```

Aufruf im Programmtext durch:

```
if hatFreiePositionen(spielstand):
        waehleEineZufaellig(gibAlleFreienPositionen(spielstand))
```

III.e
- **Stellen** Sie **dar**, wie die zufällige Auswahl im Programm umgesetzt wird. Gehen Sie dabei auch auf die Bedeutung der Funktion `hatFreiePositionen(spielstand)` ein
- **Erläutern** Sie, weshalb das beschriebene Vorgehen nicht einer realen Spielsituation entspricht.
- **Begründen** Sie, weshalb durch dieses Vorgehen das Programm trotzdem erfolgreich lernen kann, Tic-Tac-Toe zu spielen.

(12 BE)

[12] Auf die Betrachtung der Bedeutung der Farben zur Kennzeichnung eines zu wählenden freien Nachfolgefelds wird in dieser Aufgabe zur Vereinfachung verzichtet.

Anlage zur Aufgabe MENACE (Python-Version)

Anlange 1 MENACE

Der britische Informatiker und KI-Forscher Donald Michie entwickelte 1960 mit MENACE („Machine Educable Noughts And Crosses Engine“) einen „Computer“ auf Basis hunderter Streichholzschachteln, der Tic-Tac-Toe lernen konnte. In den Schächtelchen, die jeweils einen möglichen Spielstand repräsentierten, waren durch verschiedenfarbige Perlen die möglichen Züge gespeichert. Je nachdem, ob ein Spiel verloren ging oder gewonnen wurde, wurden die entsprechenden Perlen entfernt oder gleichfarbige dazugelegt. Dadurch lernte das System erfolgreiche Züge und war nach einigen hundert Partien unbesiegbar.

https://en.wikipedia.org/wiki/Matchbox_Educable_Noughts_and_Crosses_Engine, 10.08.2025 10:49

Anlage 2 Spielregeln Tic-Tac-Toe

Auf einem quadratischen, 3×3 Felder großen Spielfeld setzen die beiden Spieler abwechselnd ihr Zeichen (ein Spieler Kreuze, der andere Kreise) in ein freies Feld. Der Spieler, der als Erster drei Zeichen in eine Zeile, Spalte oder Diagonale setzen kann, gewinnt. Wenn allerdings beide Spieler optimal spielen, kann keiner gewinnen, und es kommt zu einem Unentschieden. Das heißt, alle neun Felder sind gefüllt, ohne dass ein Spieler die erforderlichen Zeichen in einer Reihe, Spalte oder Diagonalen setzen konnte.

Anlage 3 Python-Funktionen `gibFeld`, `setzeFeld`, `generiere`

```
def gibFeld(spielstand,zeilenNummer,spaltenNummer):
    '''gibt den Inhalt des Spielstands an der Position zurück'''
    ...

def setzeFeld(zeichen,spielstand,zeilenNummer,spaltenNummer):
    '''setzt den Inhalt des Spielstands an der Position
    und gibt den neuen Spielstand zurück'''
    ...

def generiere(spielstand, zeichen):
    for zeilenNummer in range(3):
        for spaltenNummer in range(3):
            spielstand=setzeFeld(zeichen,
                                 spielstand,
                                 zeilenNummer,
                                 spaltenNummer)
            if zeichen=='x':
                zeichen='o'
            else:
                zeichen='x'
    return spielstand
```

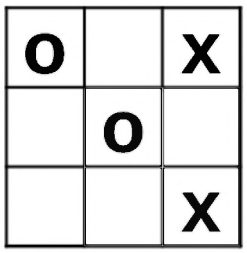
Aufgabe III: MENACE (Java-Version) 50 BE

Schwerpunkt: Intelligente Suchverfahren

Unter dem Namen MENACE hat Donald Michie in den 1960er Jahren eine Idee entwickelt, bei der er noch ohne einen Computer einzusetzen das Vorgehen moderner KI-Anwendungen vorwegnahm.

(*Text dazu und zu Tic-Tac-Toe siehe Anlagen 1 und 2*)

Für die in dem Text beschriebene Lernphase sollen in dieser Aufgabe einige Teilschritte modelliert werden.



Teilaufgaben

III.a
- **Beschreiben** Sie, wie man mit Hilfe von Breitensuche alle möglichen Spielzustände nach dem Setzen der ersten drei Zeichen (also zum Beispiel Kreis, Kreuz, Kreis) bestimmen kann.
- **Bestimmen** Sie dabei auch die Anzahl der möglichen verschiedenen Abfolgen beim Setzen der drei Zeichen.
- **Begründen** Sie, dass sich viele dieser Spielzustände prinzipiell (also im Sinn der Spielidee von Tic-Tac-Toe) nicht voneinander unterscheiden.

(12 BE)

III.b
- **Beschreiben** Sie die folgende angegebene Datenstruktur zum Verwalten des jeweiligen Spielstands von Tic-Tac-Toe. Verwenden Sie dazu das oben dargestellte Bild als Beispiel. Dort hat der Spieler 1 (Kreise) bereits zweimal gesetzt und der Spieler 2 (Kreuze) ebenfalls, so dass der Spieler 1 als nächster setzen kann.

```
char[][] beispiel = {
    {'o', ' ', 'x'},
    {' ', 'o', ' '},
    {' ', ' ', 'x'}
};
```

- **Vergleichen** Sie mit einer Verwaltung in der folgenden Form:

```
char[] beispiel
    = {'o', ' ', 'x', ' ', 'o', ' ', ' ', ' ', 'x'};
```

(10 BE)

III.c • **Erläutern** Sie, wie die im Folgenden angegebene Funktion `hatGewonnenZeile()` den aktuellen Spielstand daraufhin überprüft, ob eine der drei Zeilen mit denselben Zeichen belegt ist, so dass der jeweilige Spieler / die jeweilige Spielerin gewonnen hat.
Der jeweilige Spielstand wird in der Klasse, zu der die angegebenen Methoden gehören, in einem Attribut `spielstand` gehalten. Dazu sind hier zu Ihrer Hilfe seine Deklaration und Definition angegeben, werden aber für die Lösung der Aufgabe nicht benötigt:

```
char[][] spielstand = new char[3];
```

```
public boolean hatGewonnenZeile() {
   char[] zeile;
   for (int i=0; i<3; i++) {
       zeile = holeZeile(i);
       if (zeile[0]!=' ') {
           if (zeile[0]==zeile[1]) {
               if (zeile[0]==zeile[2]) {
                   return true;
               }
           }
       }
   }
   return false;
}

/**
 * holeZeile
 * gibt die i-te Zeile des Spielstands zurueck
*/
private char[] holeZeile(int i) {
  . . . // Code fehlt hier
}
```

• **Entwickeln** Sie das Vorgehen für eine Funktion `hatGewonnenSpalte()`.

(7 BE)

Die in der Anlage 3 angegebene Funktion generiert nicht (!) alle Möglichkeiten mit Hilfe von Tiefensuche. Die Funktion startet mit einem leeren Feld.

III.d • **Analysieren** Sie die Aufgabe der Funktion `generiere(char zeichen)` in Anlage 3.
• **Geben** Sie **an**, was sie beim Aufruf von `generiere('o')` generiert. (Es kommt dabei nicht auf die Form der Ausgabe an, sondern allein auf den sinnvollen Inhalt.)
• **Untersuchen** Sie, was in der Funktion für eine Umsetzung einer Tiefensuche fehlt.

(9 BE)

Das Vorgehen von Donald Michie soll mit dem Computer simuliert werden. Dazu führen wir zu untersuchten Abfolgen von Zügen jeweils eine Bewertung der Spielstände entsprechend dem Vorgehen von Donald Michie ein.
Dabei gehen wir so vor, dass in einer ausreichend großen Anzahl von Abfolgen von Zügen schrittweise jede so erzeugt wird, dass bei jedem Spielstand ein noch freies Feld zufällig zum Besetzen mit dem Zeichen des jeweils aktuellen Spielers ausgewählt werden soll. Mit dieser neuen Belegung soll dann jeweils der Zug fortgesetzt werden.[13]

Ein Programm zur Umsetzung dieses Vorgehens enthält die folgenden drei Funktionen:

```
/**
 * hatFreiePositionen
 * untersucht, ob es im aktuellen Spielstand
 * noch freie Positionen gibt
 */
public boolean hatFreiePositionen() {
    . . . // Code fehlt hier
}

/**
 * gibAlleFreienPositionen
 * gibt die Positionen (Array der Länge 2)
 * aller freien Felder als Array zurueck
 */
public char[][] gibAlleFreienPositionen() {
    . . . // Code fehlt hier
}

/**
 * waehleEineZufaellig
 * waehlt eine der Positionen zufaellig aus und gibt sie zurueck
 */
public char[] waehleEineZufaellig(char[][] positionen) {
    . . . // Code fehlt hier
}
```

Aufruf im Programmtext durch:

```
if hatFreiePositionen(spielstand):
        waehleEineZufaellig(gibAlleFreienPositionen())
```

III.e
- **Stellen** Sie **dar**, wie die zufällige Auswahl im Programm umgesetzt wird. Gehen Sie dabei auch auf die Bedeutung der Funktion `hatFreiePositionen(spielstand)` ein
- **Erläutern** Sie, weshalb das beschriebene Vorgehen nicht einer realen Spielsituation entspricht.
- **Begründen** Sie, weshalb durch dieses Vorgehen das Programm trotzdem erfolgreich lernen kann, Tic-Tac-Toe zu spielen.

(12 BE)

[13] Auf die Betrachtung der Bedeutung der Farben zur Kennzeichnung eines zu wählenden freien Nachfolgefelds wird in dieser Aufgabe zur Vereinfachung verzichtet.

Anlage zur Aufgabe MENACE (Java-Version)

Anlage 1 MENACE

Der britische Informatiker und KI-Forscher Donald Michie entwickelte 1960 mit MENACE („Machine Educable Noughts And Crosses Engine“) einen „Computer“ auf Basis hunderter Streichholzschachteln, der Tic-Tac-Toe lernen konnte. In den Schächtelchen, die jeweils einen möglichen Spielstand repräsentierten, waren durch verschiedenfarbige Perlen die möglichen Züge gespeichert. Je nachdem, ob ein Spiel verloren ging oder gewonnen wurde, wurden die entsprechenden Perlen entfernt oder gleichfarbige dazugelegt. Dadurch lernte das System erfolgreiche Züge und war nach einigen hundert Partien unbesiegbar.

https://en.wikipedia.org/wiki/Matchbox_Educable_Noughts_and_Crosses_Engine, 10.08.2025 10:49

Anlage 2 Spielregeln Tic-Tac-Toe

Auf einem quadratischen, 3×3 Felder großen Spielfeld setzen die beiden Spieler abwechselnd ihr Zeichen (ein Spieler Kreuze, der andere Kreise) in ein freies Feld. Der Spieler, der als Erster drei Zeichen in eine Zeile, Spalte oder Diagonale setzen kann, gewinnt. Wenn allerdings beide Spieler optimal spielen, kann keiner gewinnen, und es kommt zu einem Unentschieden. Das heißt, alle neun Felder sind gefüllt, ohne dass ein Spieler die erforderlichen Zeichen in einer Reihe, Spalte oder Diagonalen setzen konnte.

Anlage 3 Java-Funktion `generiere`

```
public void generiere(char zeichen) {
    for (int zNr=1; zNr<=3; zNr++) {
        for (int spNr=1; spNr<=3; spNr++) {
            setzeEinZeichen(zNr, spNr, zeichen);
            if (zeichen=='x') zeichen='o';
            else zeichen='x';
        }
    }
}
```