Aufgabe: Odd or Even (Leicht)

Aufgabenstellung

Ziel: Entwickle ein Programm in Java, das überprüft, ob eine Zahl gerade oder ungerade ist, und entsprechend eine Nachricht zurückgibt.

Beschreibung: Du wirst eine Methode schreiben, die eine Zahl als Eingabe nimmt und prüft, ob sie gerade oder ungerade ist. Abhängig vom Ergebnis soll die Methode einen bestimmten Text zurückgeben. Deine Implementierung wird durch vorbereitete Tests überprüft.

Schritte:

1. **Code aus dem Repository klonen:**
    - Klone das GitHub Repository, das den zu bearbeitenden Code und die Tests enthält.
2. **Methode erstellen:**
    - Öffne das Projekt in deiner bevorzugten Entwicklungsumgebung (IDE).
    - Erstelle deine Klasse und implementiere das Interface OddOrEvenGame, dies funktioniert mit dem Stichwort implements.
    - Überprüfe in der Methode, ob die Zahl gerade oder ungerade ist:
        - Eine Zahl ist gerade, wenn sie durch 2 teilbar ist (z.B. 4, 6, 8).
        - Eine Zahl ist ungerade, wenn sie nicht durch 2 teilbar ist (z.B. 3, 5, 7).
3. **Nachricht zurückgeben:**
    - Wenn die Zahl gerade ist, soll die Methode den Text "Die Zahl [zahl] ist grade." zurückgeben.
    - Wenn die Zahl ungerade ist, soll die Methode den Text "Die Zahl [zahl] ist ungrade." zurückgeben.
4. **Tests bestehen:**
    - Stelle sicher, dass alle vorbereiteten Tests im Projekt erfolgreich abgeschlossen werden.
    - Die Tests überprüfen, ob deine Methode korrekt arbeitet. Arbeite die TODOs im Testcode ab, um sicherzustellen, dass alle Tests erfolgreich sind.

Viel Spaß bei der Programmierung und beim Testen deiner Methode!

Aufgabe: Schere / Stein / Papier (Leicht)

Aufgabenstellung

Ziel: Entwickle ein Programm in Java, das überprüft, ob der Spieler in Schere / Stein / Papier gewonnen, verloren oder unentschieden gespielt hat und dementsprechend einen String ausgibt.

Schritte:

1. **Code aus dem Repository klonen:**
    - Klone das GitHub Repository, das den zu bearbeitenden Code und die Tests enthält.
2. **Methoden vervollständigen**
    - Öffne das Projekt in deiner bevorzugten Entwicklungsumgebung (IDE).
    - Nutze die Klasse „SchereSteinPapierLogik“ und implementiere die vorgegebenen Methoden. Du darfst nur den Inhalt der Methoden bearbeiten, aber nicht die Methodenköpfe.
    - Die GUI kannst du in der Klasse „SchereSteinPapierSpiel“ starten.
3. **Ist Gewinner Spieler**
    - Es soll überprüft werden, ob der Spieler gewonnen hat. Dafür musst du die beiden Strings: „playerChoice“ und „robotChoice“, miteinander vergleichen. Wenn der Spieler gewonnen hat, muss: „true“, zurückgegeben werden, ansonsten: „false“. (Roboter gewinnt oder unentschieden)
4. **Bestimme Gewinner String**
    - Auf der GUI soll der Gewinner String angezeigt werden. Dort soll entweder „Du gewinnst!“, „Roboter gewinnt!“ oder „Unentschieden!“ stehen. Du musst anhand der: „playerChoice“ und „robotChoice“, den entsprechenden String zurückgeben.
    - Tipp: Du kannst auch die Methode: „istGewinnerSpieler“, nutzen.
5. **Bestimme Ergebnis String**
    - Auf der Oberfläche soll außerdem stehen, was der Spieler und was der Roboter gewählt hat. Baue dafür einen String zusammen und gib ihn mit return zurück an die Oberfläche. Der String muss zuerst den „playerChoice“ und dann den „robotChoice“ String beinhalten. Beispiel: „Spieler: STEIN | Roboter: SCHERE“.
6. **Tests bestehen**
    - Stelle sicher, dass alle vorbereiteten Tests im Projekt erfolgreich abgeschlossen werden.

Viel Erfolg bei der Programmierung und beim Testen deiner Methoden!

Aufgabe: FizzBuzz (Leicht)

Aufgabenstellung

Ziel: Gebe alle Zahlen von 1 bis 100 aus und ersetze jedes unter bestimmten Kriterien die Zahl durch einen Text.

Schritte:

1. **Code aus dem Repository klonen:**
    - Klone das GitHub Repository, das den zu bearbeitenden Code enthält.
2. **Methoden vervollständigen**
    - Öffne das Projekt in deiner bevorzugten Entwicklungsumgebung (IDE).
    - Nutze die Klasse „Main“. Du darfst nur den Inhalt der Methoden bearbeiten, aber nicht die Methodenköpfe
3. **Zahlen ausgeben**
    - Gebe die Zahlen zwischen 1 und 100 an der Konsole aus. Beachte dabei das ein Computer ab 0 anfängt zu Zählen
4. **Zahlen ersetzen**
    - Jetzt musst du jedes Mal, wenn eine Zahl an der Konsole ausgegeben werden soll, welche durch 3 teilbar ist, den Text “Fizz” anstatt der Zahl ausgeben. Danach soll es normal weitergehen. Nun gebe auch den Text “Buzz” aus, wenn die Zahl, die ausgegeben werden soll, durch 5 teilbar ist.
      Das sollte etwa so aussehen:
      1 2 Fizz 4 Buzz Fizz 7 8 9 Buzz 11 Fizz 13 14 ...
5. **FizzBuzz**
    - Was ist aber wenn eine Zahl durch 3 und 5 teilbar ist? Dann soll anstatt der Zahl der Text “FizzBuzz” ausgegeben werden. Also so:
      1 2 Fizz 4 Buzz Fizz 7 8 9 Buzz 11 Fizz 13 14 FizzBuzz 16 ...

Viel Erfolg bei der Programmierung und beim Testen deiner Methoden!

Aufgabe: Sieb des Eratosthenes (Mittel)

Aufgabenstellung

Ziel: Entwickle ein Programm in Java, das die Primzahlen von 1 bis n mit dem vorgegebenen Algorithmus berechnet und zurückgibt.

Schritte:

1. **Code aus dem Repository klonen:**
    - Klone das GitHub Repository, das den zu bearbeitenden Code und die Tests enthält.
2. **Methoden vervollständigen**
    - Öffne das Projekt in deiner bevorzugten Entwicklungsumgebung (IDE).
    - Nutze die Klasse „SiebDesEratosthenes“ und vervollständige die Implementierung. Du kannst dafür die vorgegebenen Methoden benutzen, oder alles löschen und von vorne anfangen. Die Methode „berechnePrimzahlen“ muss so bleiben.
    - Der Algorithmus funktioniert nach dem folgenden Prinzip
        1. Man schreibe alle Zahlen von 1 bis n hin und streiche die Zahl 1 durch.
        2. Sei i die kleinste noch nicht durchgestrichene und nicht eingerahmte Zahl. Solange i existiert und i^2 ≤ n ist, rahme man i ein und streiche alle Vielfachen von i durch (die Vielfachen von i werden „ausgesiebt“).
        3. Die eingerahmten und die nicht durchgestrichenen Zahlen sind die Primzahlen von 1 bis n.
3. **Array und Liste erstellen (vorgegeben)**
    - Implementiere die Methode „erstelleArrayUndListe“. Das ist der erste Schritt des Algorithmus. Es soll die Liste „primzahlen“ erstellt werden und der Array „zahlen“, der zusätzlich mit allen Zahlen von 1 bis n gefüllt werden soll.
4. **Durchstreiche Vielfache (vorgegeben)**
    - Implementiere die Hilfsmethode „durchstreicheVielfache“. Sie soll die Vielfachen einer Zahl bis n durchstreichen.
5. **Sieben (vorgegeben)**
    - Implementiere die Methode „sieben“. Das ist der zweite Schritt des Algorithmus. Implementiere ihn so wie angegeben. Benutze dafür die Methoden „istDurchgestrichen“ und „durchstreicheVielfache“.
6. **Primzahlen rausschreiben (vorgegeben)**
    - Implementiere die Methode „primesRausschreiben“. Das ist der letzte Schritt des Algorithmus. Hänge alle Zahlen an die „primzahlen“ Liste an, die nicht gestrichen sind.
7. **Test bestehen**
    - Stelle sicher, dass alle vorbereiteten Tests im Projekt erfolgreich abgeschlossen werden.

Viel Erfolg bei der Programmierung und beim Testen deiner Methoden!

Aufgabe: Palindrom (Mittel)

Aufgabenstellung

Ziel: Entwickle ein Programm in Java, welches als Eingabe eine Zahl bekommt und überprüft, ob diese eine Palindrom ist

Schritte:

1. **Code aus dem Repository klonen:**
    - Klone das GitHub Repository, das den zu bearbeitenden Code und die Tests enthält.
2. **Methoden vervollständigen**
    - Öffne das Projekt in deiner bevorzugten Entwicklungsumgebung (IDE).
    - Nutze die Klasse „Main“ und vervollständige die Implementierung.
    - Fasse nur die “Main” klasse an, der Rest soll so bestehen bleiben.
3. **Algorithmus**
    - Der Algorithmus soll überprüfen, ob eine gegebene Zahl ein Palindrom ist. Ein Palindrom ist eine Zahl die vorwärts und rückwärts gleich geschrieben werden
    - 4235 ist kein Palindrom da die Zahl rückwärts 5324 geschrieben wird
    - 6336 ist ein Palindrom, da die Zahl rückwärts 6336 geschrieben wird
4. **Eingabe**
    - Du kannst Ermstal die Zahl fest im Code setzen. Nachdem du den Algorithmus implementiert hast, kannst du dich daransetzen eine Eingabe über die Konsole zu erstellen.
5. **Test bestehen**
    - Stelle sicher, dass alle vorbereiteten Tests im Projekt erfolgreich abgeschlossen werden.

Viel Erfolg bei der Programmierung und beim Testen deiner Methoden!

Aufgabe: Tic-Tac-Toe Spiel (Schwer)

Aufgabenstellung

Ziel: Entwickle ein Java-Programm, das ein Tic-Tac-Toe Spiel implementiert. Die Methode muss Züge ausführen, den Gewinner ermitteln und das Spiel zurücksetzen können. Deine Implementierung wird durch vorbereitete Tests überprüft.

Schritte:

1. **Code aus dem Repository klonen:**
    - Klone das GitHub Repository, das den zu bearbeitenden Code und die Tests enthält.
2. **Methoden implementieren:**
    - Öffne das Projekt in deiner bevorzugten Entwicklungsumgebung (IDE).
    - Finde die Klasse TicTacToeGame und implementiere die fehlenden Methoden makeMove(int index), getWinner(), und resetGame().
3. **Zug implementieren:**
    - Implementiere die Methode makeMove(int index) so, dass ein Zug korrekt ausgeführt wird. Berücksichtige, dass das Feld nur besetzt werden kann, wenn es frei ist, und wechsle zwischen den Spielern X und O.
4. **Gewinner überprüfen:**
    - Implementiere die Methode getWinner() so, dass die Überprüfung der Zeilen, Spalten und Diagonalen korrekt durchgeführt wird. Wenn ein Spieler drei Felder in einer Reihe, Spalte oder Diagonale besetzt hat, ist dieser Spieler der Gewinner.
    - Überprüfe auch, ob das Spiel unentschieden ist (kein freies Feld mehr, aber kein Gewinner).
5. **Spiel zurücksetzen:**
    - Implementiere die Methode resetGame(), um das Spiel auf den Anfangszustand zurückzusetzen (alle Felder leer und der letzte Spieler auf O).

Anforderungen:

- Implementiere die Klasse TicTacToeGame und die Methoden makeMove(int index), getWinner(), und resetGame() so, dass alle Tests erfolgreich bestehen.
- Achte darauf, dass die Spielregeln korrekt umgesetzt werden.

Viel Erfolg bei der Programmierung und beim Testen deiner Methoden!

Java Cheat Sheet

Basisdatentypen

- int – 32-bit Ganzzahl
- double – 64-bit Gleitkommazahl
- boolean – Wahrheitswert (true/false)
- char – 16-bit Unicode-Zeichen

Variablen definieren

- int zahl = 10;
- double pi = 3.14;
- boolean istWahr = true;

Operatoren

Arithmetische Operatoren

- + – Addition
- - – Subtraktion
- * – Multiplikation
- / – Division
- % – Modulo (Rest)

Vergleichsoperatoren

- == – Gleichheit
- != – Ungleichheit
- < – Kleiner als
- > – Größer als
- <= – Kleiner oder gleich
- >= – Größer oder gleich

Logische Operatoren

- && – Logisches UND
- || – Logisches ODER
- ! – Logisches NICHT

Kontrollstrukturen

Bedingte Anweisungen

- if (bedingung) {
    - // Code, wenn Bedingung wahr ist
    - } else if (andereBedingung) {
        - // Code, wenn andere Bedingung wahr ist
        - } else {
            - // Code, wenn keine der Bedingungen wahr ist
            - }

Schleifen

- for (int i = 0; i < 10; i++) {

  // Code

  }
- while (bedingung) {

  // Code

  }
- do {

  // Code

  } while (bedingung);

Arrays

- int[] zahlen = new int[10];
- int[] andereZahlen = {1, 2, 3, 4, 5};
- zahlen[0] = 42; // Zugriff auf ein Element

Methoden

Deklaration

- public Rückgabewert methodenName(Typ parameter) {

  // Code

  return wert;

  }

Aufruf

- int ergebnis = methodenName(argument);

Objekt erstellen

- MeineKlasse objekt = new MeineKlasse(10);

Eingabe und Ausgabe

Konsoleneingabe

- import java.util.Scanner;
- Scanner scanner = new Scanner(System.in);
- System.out.print("Geben Sie einen Wert ein: ");
- int eingabe = scanner.nextInt();

Konsolenausgabe

- System.out.println("Hallo, Welt!");

Listen

- ArrayList<String> liste = new ArrayList<>();
- liste.add("Element");
- String element = liste.get(0);

Maps

- HashMap<String, Integer> map = new HashMap<>();
- map.put("Schlüssel", 42);
- int wert = map.get("Schlüssel");
