Aufgabenblatt 4: Algorithmen implementieren II

Alle Aufgaben auf diesem Blatt sind mit der Entwicklungsumgebung BlueJ zu bearbeiten.

Aufgabe 1[1]

Eine gute Möglichkeit, um erstes Arbeiten mit Methoden und Kontrollstrukturen zu üben, stellen Turtlegrafiken dar, die sowohl in der bereits bekannten Programmierumgebung Greenfoot als auch in BlueJ genutzt werden können. Bevor Sie mit dem Programmieren starten können, gehen Sie bitte auf `http://www.java-online.ch/lego/turtleGrafik` und laden sich die Klassenbibliothek `aplu5.jar` herunter. Integrieren Sie diese in BlueJ folgendermaßen:

- Starten Sie BlueJ.
- Im Menü oben gehen Sie auf Werkzeuge (Windows / Linux) bzw. auf BlueJ (OS) → Einstellungen → Bibliotheken → Hinzufügen aplu5.jar auswählen → ok → ok
- Reset der Virtuellen Maschine mit Rechtsklick auf den Balken unten rechts oder über Menüpunkt.

Lesen Sie sich auf der oben verlinkten Website in die Anwendung der Turtlegrafik ein und bearbeiten Sie dann die folgenden Teilaufgaben:

(a) Implementieren Sie unter Verwendung der Wiederholung mit fester Anzahl die Methode `sechseckZeichnen()`.

Lösungsvorschlag

```
private void sechseckZeichnen() {
  for (int i = 0; i < 6; i++) {
    turtle.forward(50);
    turtle.right(60);
  }
}
```

[1] *Qualifizierungsmaßnahme Informatik: Objektorientierte Modellierung und Programmierung: Aufgabenblatt 4: Algorithmen implementieren II.*

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/turtle/Blume.java`

(b) Ergänzen Sie Ihr Projekt um eine Methode `blumeZeichnen()`, die mit Hilfe der Methode `sechseckZeichnen()` folgende Zeichnung erstellt:

Lösungsvorschlag

```java
public void blumeZeichnen() {
  int anzahl = 20;
  double drehung = 360 / anzahl;
  turtle.speed(1000);
  for (int i = 0; i < anzahl; i++) {
    sechseckZeichnen();
    turtle.right(drehung);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/turtle/Blume.java`

(c) Zeichnen Sie eine Europa-Flagge, indem Sie zuerst eine Methode `star()`, die einen Stern zeichnet, schreiben. Die blaue Hintergrundfarbe erhält man mit der Methode `clear(Color.blue)`.

```
public class EuropaFlagge {
  Turtle turtle;

  public EuropaFlagge() {
    turtle = new Turtle();
    turtle.hideTurtle();
    turtle.penUp();
    turtle.clear(Color.blue);
  }

  private void zeichneStern() {
    int zacken = 5;
    int winkel = 60;
    int größe = 8;
    turtle.setPenColor(Color.yellow);
    turtle.penDown();
    int ausgleichsWinkel = 360 / zacken + winkel;
    turtle.fillToPoint(turtle.getX(), turtle.getY()) ;
    for (int i = 0; i < zacken; i++) {
      turtle.forward(größe);
      turtle.right(ausgleichsWinkel);
      turtle.forward(größe);
      turtle.left(winkel);
    }
    turtle.fillOff();
```

```
    turtle.penUp();
  }

  public void zeichne() {
    double anzahl = 27;
    double radius = 170;
    double rotation = 360 / anzahl;
    for (int i = 0; i < anzahl; i++) {
      turtle.forward(radius);
      zeichneStern();
      turtle.back(radius);
      turtle.right(rotation);
    }
  }

  public static void main(String[] args) {
    EuropaFlagge flagge = new EuropaFlagge();
    flagge.zeichne();
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/turtle/EuropaFlagge.java`

Die vorangegangenen Aufgabenstellungen und die verwendete Bibliothek sind dem Material des Forschungsprojekts PHBern mit freundlicher Genehmigung durch Frau Jarka Arnold entnommen.

Aufgabe 2[2]

Modellierung und Implementierung des Spiels *„No Risk No Money“*

Spielidee:

- Der Spieler kann so oft würfeln, wie er will.
- Mit jedem Wurf erhöht sich sein Gewinn um eins, sofern die gewürfelte Zahl nicht vorher schon einmal gewürfelt wurde.
- Wenn man eine Zahl das zweite Mal würfelt, ist der Gewinn weg.

[2] *Qualifizierungsmaßnahme Informatik: Objektorientierte Modellierung und Programmierung: Aufgabenblatt 4: Algorithmen implementieren II*.

Ablauf:

Der Spieler würfelt einmal. Hat er eine Zahl gewürfelt, die noch nicht gewürftelt wurde, erhöht sich sein Gewinn um eins. Dies wiederholt der Spieler so oft er möchte. Er kann sich jederzeit ausgeben lasse, welche Zahlen er bereits gewürfelt hat.

Spielende:

Der Spieler kann entscheiden wie oft er spielen möchte. Hört er auf, obwohl er noch spielen könnte, dann darf er den Gewinn behalten. Würfelt er eine Zahl das zweite mal, dann ist das Spiel beendet und der Spieler erhält keinen Gewinn.

(a) Begründen Sie, weshalb sich zur Verwaltung der bereits gewürfelten Zahlen ein Feld vom Typ `boolean` eignet.

(b) Erstellen Sie eine zur obigen Spielbeschreibung passende Klassenkarte.

(c) Implementieren Sie das Spiel wie oben beschrieben und unter Beachtung der Modellierung aus Teilaufgabe (b). Dabei können Sie zunächst davon ausgehen, dass immer mit einem normalen *„sechsseitigen“* Würfel gewürfelt wird. Achten Sie darauf, dass dem Spieler immer passende Ausgaben zum aktuellen Spielstand angezeigt werden.

(d) Beschreiben Sie in kurzen Sätzen, welche Fälle beim Testen des Spieles beachtet werden müssen, und testen Sie ihr Spiel danach ausführlich.

(e) Ändern Sie die Implementierung des Spieles so ab, dass der Spieler beim Start selbst entscheiden kann, mit welcher Art Würfel er spielen möchte. Dabei muss er mindestens einen vierseitigen Würfel wählen.

Lösungsvorschlag

```
public class Spiel {
  private int gewinn;
  private boolean[] würfe;
  private boolean spielVerloren;
  private boolean spielBeendet;
  private int würfelSeitenAnzahl;

  public Spiel() {
```

```
    würfelSeitenAnzahl = 6;
    spielBeginnen();
  }

  public Spiel(int würfelSeitenAnzahl) {
    this.würfelSeitenAnzahl = würfelSeitenAnzahl;
    spielBeginnen();
  }

  private int gibWürfelZufallsZahl() {
    return (int) (Math.random() * würfelSeitenAnzahl) + 1;
  }

  public void spielBeginnen() {
    gewinn = 0;
    spielBeendet = false;
    spielVerloren = false;
    würfe = new boolean[würfelSeitenAnzahl];
  }

  public void würfle() {
    if (spielVerloren) {
      System.out.println(
        "Sie können nicht mehr würfeln, "+
        "weil Sie das Spiel bereits verloren haben.");
      return;
    }
    if (spielVerloren) {
      System.out.println(
          "Sie können nicht mehr würfeln, weil Sie das Spiel " +
          "bereits beendet haben. Starten sie ein neues Spiel");
      return;
    }
    int wurf = gibWürfelZufallsZahl();
    System.out.println("Sie haben " + wurf + " gewürfelt.");

    if (!würfe[wurf - 1]) {
      gewinn++;
      würfe[wurf - 1] = true;
      System.out.println("Sie können weiterspielen. " +
        "Ihr aktueller Punktestand ist " + gewinn + " Punkte.");
      return;
    }
    spielVerloren = true;
    System.out.println("Sie haben verloren. Sie haben die Zahl " +
      "schon einmal gewürfelt.");
  }
```

```java
  public void berichteÜberWürfe() {
    for (int i = 0; i < würfe.length; i++) {
      boolean wurf = würfe[i];
      int zahl = i + 1;
      if (wurf) {
        System.out.println("Die Zahl " + zahl +
          " wurde bereits geworfen.");
      } else {
        System.out.println("Die Zahl " + zahl +
          " wurde noch nicht geworfen.");
      }
    }
  }

  public void spielBeenden() {
    if (spielBeendet) {
      System.out.println("Sie haben das Spiel bereits beendet.");
      return;
    }
    spielBeendet = true;
    if (spielVerloren) {
      System.out.println("Sie haben leider verloren");
    } else {
      System.out.println("Sie haben gewonnen. " +
        "Ihr Punktestand lautet: " + gewinn + ".");
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/risk/Spiel.java`

Diese Aufgabe entstammt den Materialien von Inf-schule.de. Diese sind unter Creative Commons BY-SA 4.0 lizenziert. Das ursprüngliche Material wurde in der vorliegenden Aufgabe adaptiert.

Aufgabe 3 (Check-Up)[3]

Das Kino der Kleinstadt Cinehausen möchte die Kartenreservierung ab sofort digital ermöglichen. Es besteht aus 5 Kinosälen, die jeweils 180 Sitzplätze beinhalten. Jeden Tag gibt es pro Kinosaal nur eine Vorstellung um 20.15 Uhr. Die gezeigten Filme können in den Sälen unterschiedlich sein, so dass Besucher bei der Reservierung angeben müssen, welchen Sitzplatz sie in welchem Saal buchen möchten. Die Reservierung ist immer nur für den

[3] *Qualifizierungsmaßnahme Informatik: Objektorientierte Modellierung und Programmierung: Aufgabenblatt 4: Algorithmen implementieren II.*

gleichen Tag möglich. Jeden Tag starten die vergebenen Buchungsnummern wieder bei 1 und werden dann nach jeder vorgenommenen Reservierung um eins erhöht. Wenn das Ticket vom Besucher abgeholt wurde, wird die Buchungsnummer des Sitzplatzes gelöscht.

Weitere Informationen über die gewünschte Modellierung können Sie dem folgenden Klassendiagramm entnehmen:

Setzen Sie das gegebene Szenario passend in einem BlueJ-Projekt um. Schreiben Sie außerdem eine Testklasse, die alle geforderten Methoden ausgiebig testet. Dabei sollen auch Fälle abgefangen werden, bei denen zum Beispiel bei der Ticketabholung eine ungültige bzw. nicht vorhandene Buchungsnummer angegeben wird oder ein Sitzplatz reserviert werden soll, der bereits vergeben oder im Kino nicht vorhanden ist. Achten Sie auch darauf, dass dem Benutzer passende Textausgaben angezeigt werden.

```java
public class Kino {
  private int buchungsNrZaehler;
  private Kinosaal[] kinosaele;
  private Reservierung[] reservierungen;

  public Kino() {
    kinosaele = new Kinosaal[5];
    reservierungen = new Reservierung[5 * 180];
    for (int i = 0; i < 5; i++) {
      kinosaele[i] = new Kinosaal(i + 1);
    }
    buchungsNrZaehler = 0;
  }

  public void sitzplatzReservieren(int saalNr, int sitzplatzNr) {
    Kinosaal kinosaal = kinosaele[saalNr - 1];
    boolean ergebnis = kinosaal.sitzplatzReservieren(sitzplatzNr,
    ↪  buchungsNrZaehler + 1);
    if (ergebnis) {
      buchungsNrZaehler++;
      System.out.println("Ihre Reservierungsnummer lautet: " +
      ↪  buchungsNrZaehler);
      reservierungen[buchungsNrZaehler - 1] = new
      ↪  Reservierung(kinosaal,
      ↪  kinosaal.gibSitzplatzBeiNummer(sitzplatzNr),
          buchungsNrZaehler);
    } else {
      System.out.println(
          "Der gewünschte Sitzplatz mit der Nummer " + sitzplatzNr +
          ↪  " im Kino " + saalNr + " ist bereits vergeben.");
    }
```

```
  }

  public void ticketAbholen(int buchungsNr) {
    if (reservierungen[buchungsNr - 1] instanceof Reservierung) {
      Reservierung reservierung = reservierungen[buchungsNr - 1];
      reservierung.sitzplatz.besetzen();
      System.out.println("Genießen Sie Ihren Film im Kino " +
      ↪  reservierung.kinosaal.nummer + " auf dem Sitzplatz "
         + reservierung.sitzplatz.nummer + ".");
      reservierungen[buchungsNr - 1] = null;
    } else {

      ↪  System.out.println("Uns liegt keine Reservierung mit der Nummer "
      ↪  + buchungsNr + " vor.");
    }
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/kino/Kino.java`

```
/**
 * Beschreiben Sie hier die Klasse Kinosaal.
 *
 * @author (Hermine Bschlangaul)
 * @version (2020-02-05)
 */
public class Kinosaal {
  private int sitzplatzAnzahl = 180;
  private Sitzplatz[] sitzplaetze;
  public int nummer;

  public Kinosaal(int nummer) {
    this.nummer = nummer;
    sitzplaetze = new Sitzplatz[sitzplatzAnzahl];
    for (int i = 0; i < sitzplatzAnzahl; i++) {
      sitzplaetze[i] = new Sitzplatz(i + 1);
    }
  }

  public Sitzplatz gibSitzplatzBeiNummer(int sitzplatzNummer) {
    return sitzplaetze[sitzplatzNummer - 1];
  }

  /**
   * @param sitzplatzNr Die Sitzplatznummer.
   * @param buchungsNr Die Buchungsnummer.
   *
   * @return Wahr bei erfolgreicher Buchung, ansonsten falsch.
```

```
   */
  public boolean sitzplatzReservieren(int sitzplatzNr, int buchungsNr)
  ↪ {
    if (sitzplatzNr > 180 || sitzplatzNr < 1) {
      System.out
          .println("Leider besitzt der Kinosaal " + nummer +
          ↪ " keinen Sitzplatz mit der Nummer " + sitzplatzNr +
          ↪ ".");
      return false;
    }
    Sitzplatz sitzplatz = sitzplaetze[sitzplatzNr - 1];
    return sitzplatz.reservieren(buchungsNr);
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/kino/Kinosaal.java`

```
public class Reservierung {

  public Sitzplatz sitzplatz;
  public Kinosaal kinosaal;
  public int nummer;

  public Reservierung(Kinosaal kinosaal, Sitzplatz sitzplatz, int
  ↪ nummer) {
    this.kinosaal = kinosaal;
    this.sitzplatz = sitzplatz;
    this.nummer = nummer;
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/kino/Reservierung.java`

```
public class Sitzplatz {
  private boolean reserviert;
  private int buchungsNr;
  public int nummer;

  public Sitzplatz(int nummer) {
    buchungsNr = 0;
    reserviert = false;
    this.nummer = nummer;
  }

  public boolean reservieren(int buchungsnummer) {
    if (reserviert || buchungsNr > 0) {
      return false;
```

```
    }
    buchungsNr = buchungsnummer;
    return true;
  }

  public void besetzen() {
    buchungsNr = 0;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/oomup/ab_4/kino/Sitzplatz.java`