Java für Fortgeschrittene

Teil 2

Fragen zur letzten Stunde

- Nennen Sie alle Anweisungen die Sie kennen.
- Was bedeutet private, protected und public ?
- Wozu dienen die Schlüsselwörter this, super, final und static
- Erstellen sie eine Klasse namens „Car“. Geben sie Ihr zwei private Variablen für die max. Geschwindigkeit und den Hersteller Namen. Verwenden Sie sinnvolle Datentypen. Mit Hilfe zweier Parameter werden im Konstruktor die Variablen gesetzt. Erstellen Sie zwei Methoden, die für externe Klassen zugänglich sind, welche die Werte der beiden Variablen zurückgeben.
- Erstellen Sie eine Klasse „BMW“ und vererben ihr die Klasse „Car“. Im Konstruktor wird die max. Geschwindigkeit über einen Parameter belegt und der Name auf BMW. Des Weiteren soll der Konstruktor in der Konsole den Text „BMW erschaffen mit <max. Geschwindigkeit> km/h maximal Geschwindigkeit“ ausgeben.
- Erstellen Sie anschließend eine Klasse Manufactory mit der Methode public static void main(String[] str) welche einen BMW erzeugt.

Lösung der Hausaufgabe

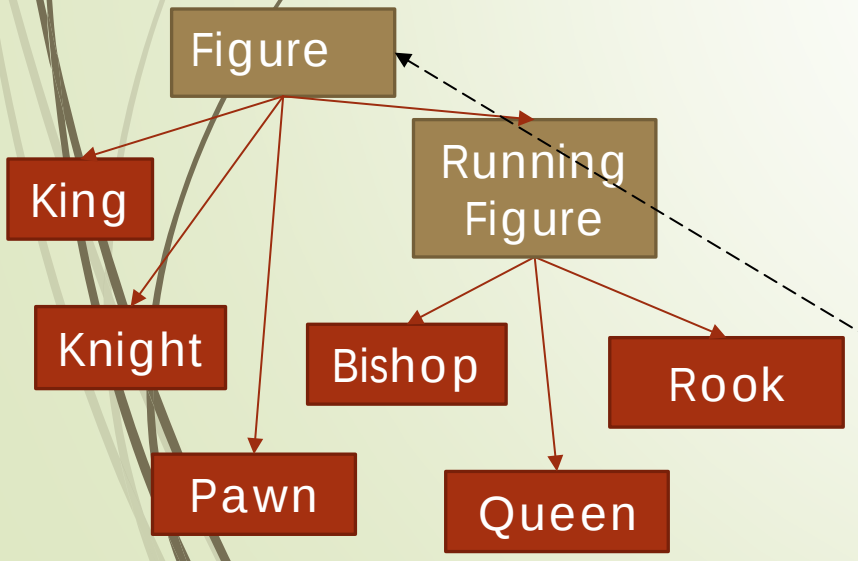
Class Chess:
Enthält Button zum Starten des Spiels.
Enthält Main-Methode, in der das Programm startet.
Ist für die Darstellung des Spiels zuständig.

Class Chessboard extends Jpanel implements IChessboard:
Methoden: startGame: Legt die Anfangsposition der Figuren fest, nextTurn,
Variablen: turn, status, selectedFigure, selectedField, *chessboard*

HashMap fields

Chessfield extends Jpanel (Value in HashMap): kann eine Figur fassen. Dient zur Visuellen Darstellung eines Feldes und ggf. einer Schachfigur. Reagiert auf Mouse-events.
Methoden: movefigureToThisField, isMyFigureOnField, isOppositeFigureOnField, isFigureOnField

Class Position (Key in HashMap):
int x, int y: Dient zum Festlegen der Position der Felder und der Figuren




ERWIN LÖST MIT DEM COMPUTER ALLE PROBLEME, DIE ER NOCH NICHT HATTE, ALS ER NOCH KEINEN COMPUTER HATTE...

Abstrakte Klassen und Methoden

- Bei abstrakten Klassen handelt es sich um Klassen, von denen man kein Objekt erzeugen kann.
- In einer abstrakten Klasse lassen sich auch abstrakte Methoden definieren, diese müssen im Erben überschrieben werden.
- Definieren lassen sich solche Klassen und Methoden mit dem Signalwort „abstract“.

http://openbook.galileocomputing.de/javainsel/javainsel_05_012.html#dodtpd2ed3e4a-d795-4c52-9357-cb640e8f256f

Interfaces

Ein Interface ist ein Container für Methoden (Methoden können in ihr definiert werden es kann aber keine Logik hinzugefügt werden). Ein Interface kann nach Erstellung in eine Klasse implementiert werden. Dort **müssen** nun alle Interface-Methoden überschrieben werden:

```
public interface IChess {
    public void status(Status status);
}
```

Implementierung in eine Klasse:

```
public class Chess extends JFrame implements IChess {
    @Override
    public void status(Status status) {
    ...
}}
```

In einem Interface deklarierte Variablen sind automatisch static und final d. h. Konstanten.

http://openbook.galileocomputing.de/javainsel/javainsel_05_013.html#dodtp4cdf05f9-ab55-4791-9fa2-c6929cc1a19d

Achtung!

Es ist zwar **nicht** möglich ein Objekt einer abstrakten Klasse oder eines Interface zu Erstellen. Es ist jedoch möglich, abstrakte Klassen und Interfaces als Typ einer Variable zu verwenden.

Nun lass uns die Variable mit dem Typ einer Klasse als Zeiger sehen. So kann er auf ein Objekt der eigenen Klasse zeigen oder auf einen der Erben. Es lassen sich jedoch nur Methoden und Variablen der eigenen/basis-Klassen verwende.

Nun lass uns die Variable mit dem Typ eines Interface als Zeiger sehen. So kann dieser auf jedes Objekt zeigen dessen Klasse das jeweilige Interface implementiert hat.

Innere Klassen

Bei Inneren Klassen handelt es sich um Klassen, die innerhalb einer anderen Klasse definiert werden:

```
public class Chessboard extends JPanel implements IChessboard {

...

public class Chessfield extends JPanel {

    ...
    }
}
```

Der Aufruf in einer Externen Klasse sieht folgender Maßen aus (Vorausgesetzt der Konstruktor Chessfield ist public):

```
Chessboard chessboard;

chessboard.new Chessfield(...);
```

http://openbook.galileocomputing.de/javainsel/javainsel_07_001.html#dodtp0b960ace-1146-4436-b4f6-3f9325e8fd1f

Generics

Generics dienen hierzu, den Typ einer Variable zur Compile-Zeit zu bestimmen:

```
public class Generic <TYPE> {

private TYPE value;

public TYPE getValue(){
return value;
}
}
```

Verwendung:

```
Generic<Integer> g1 = new Generic<Integer>(); // TYPE = Integer
Generic<String> g2 = new Generic<String>(); // TYPE = String
```

http://openbook.galileocomputing.de/javainsel/javainsel_09_001.html#dodtp1e5aa8bf-acb7-4f0f-b3cf-176e06f73845

Listen

- Objekte des selben Typs können in einer Liste gespeichert werden. Es gibt zwei Arten von Listen: LinkedList und ArrayList. Beide besitzen die selben Methoden (da beide das Interface List implementieren). Haben für den User auch gleiche Funktionalität. Sie sind Intern jedoch anders Organisiert.
- Eine LinkedList ist besonders gut wenn man die Liste häufig Sortieren muss, oder in die Liste weitere Elemente einfügen muss.
- Eine ArrayList ist sehr gut wenn man häufig über eine Liste iterieren muss.
- Des Weiteren gibt es auch einen Set. Der Set verhält sich wie eine Liste, es lassen sich jedoch jedes Element nur einmal anhängen.
- Erzeugung: List<Element> list = new ArrayList<>();
- Element Hinzufügen: list.add(element);
- Element Löschen: list.remove(element);

HashMap

Die Klasse HashMap<Key,Value> ermöglicht es einen Objekte des Typs Value unter Zuordnung zu einem Schlüssel des Typs Key in der Map festzuhalten.

- Erzeugen einer Neuen HashMap:

```
HashMap<Position, Chessfield> fields = new HashMap<>();
```

- Hinzufügen eines neuen Elements:

```
fields.put(pos, field);
```

- Element aus der Map entfernen:

```
public V remove(Object key)
```

- Auf Element mit Hilfe des Schlüssels zugreifen:

```
fields.get(new Position(1, 1));
```

Die Klasse Object

Alle Klassen die nicht von einer persönlich bestimmten Klasse erben, erben von Object. Diese Klasse enthält folgende Methoden:

- public String toString(): Gibt das Objekt als String zurück
- public boolean equals(object o): Vergleicht sich mit dem übergebenen Objekt und gibt true zurück wenn es sich um das selbe Objekt handelt.

  Standard Implementierung:

```
public boolean equals(Object obj){
        return this == obj;
}
```

- protected Object clone(): Diese Methode ist zum Klonen/Kopieren des Objekts zuständig
- public int hashCode(): gibt für jedes Objekt einen einmaligen Integer zurück.
- Hinweis: Um ein Klasse als Key für eine HashMap verwendet zu werden empfielt es sich die Methoden hashCode() und equals zu überschreiben.

http://openbook.galileocomputing.de/javainsel/javainsel_08_003.html#dodtpe2738381-3e34-44c9-9fea-fadba7f7b82d

Hausaufgabe:

- Erstellen Sie die Schachfiguren Pawn und Knight (Bauer und Springer)
- Ergänzen Sie die Klasse „Car“ um einen um einen Enum Namens WheelPosition mit den Werten FRONT_LEFT, FRONT_RIGHT, BACK_LEFT, BACK_RIGHT
- Ergänzen Sie die Klasse „Car“ um eine HashMap. Geben sie als Key den Enum WheelPosition an und als Value die Klasse Car.
- Ergänzen Sie die Klasse „Car“ um eine Interne Klasse „Wheel“, als Eigenschaft Enthält dieser die „WheelPosition“. Diese Eigenschaft
- Fügen Sie der Map im Konstruktor Car() die vier Reifen hinzu, die ein Auto haben sollte.
- Installieren Sie sich zuhause schon einmal den WindowBuilder und „Spielen“ damit.