Kontrollfragen: Objekt-orientierte Programmierung - Vererbung und Polymorphie

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B
Version: 1.0

Kontrollfragen

1. ## Ober- und Unterklassen
   1. `class X extends Y { ... }` - Welches ist die Oberklasse und welches die Unterklasse?
   2. `class Z extends Y { ... }` - Ist Z eine Unterklasse von X?
2. ## Statischer und dynamischer Typ

   Gegeben sei folgender Code:

   ```
   class X extends Y { ... }
   class Z extends Y { ... }
   Y y = new X();
   ```

   1. Welches ist der statische und welches der dynamische Typ von `y`?
   2. Kann ich `y` auch mit einem Objekt vom Typ Z initialisieren?
   3. Kann ich einer Referenzvariablen `Z z;` ein Objekt vom Typ `X` zuweisen?
   4. Kann ich einer Referenzvariablen `Z z;` ein Objekt vom Typ `Y` zuweisen?
   5. Wie teste ich, wenn ich eine Referenzvariable `Y y;` habe, ob das Objekt, auf das `y` zeigt, vom Typ `X` ist?
   6. Was passiert wenn meine Referenzvariable vom Typ `Y y` mit `null` initialisiert ist, und ich einen Typtest auf `X` durchführe?
3. ## Methoden
   1. Wann kann ich Methoden in einer Subklasse überschreiben?
   2. Was ist der Unterschied zwischen *Method Overloading* und *Method Overriding*?
   3. Was ist der Unterschied zwischen einem Konstruktor und einer Methode?
   4. Wie kann ich gezielt eine Methode der Superklasse in einer Subklasse aufrufen?
   5. Wie kann ich gezielt einen anderen Konstruktor der selben Klasse aufrufen?
   6. Welche Methoden hat jede Klasse und warum?
4. ## Überschriebene Methoden

   Gegeben sei folgender Code:

   ```
   class Y { void p(){println("Y.p");} }
   class X extends Y { void p(){println("X.p");} }
   class Z extends Y {
      void p(){println("Z.p");}
      void m(){println("Z.m");} }
   Y x = new X(); Y z = new Z();
   ```

   1. Was wird ausgegeben bei `x.p();`?
   2. Was gibt `x.p();` aus, wenn die Methode `p` in der Klasse `X` nicht überschrieben

worden wäre?
3. Wie kann ich die Methode `m` von `Z` auf der Variable `z` aufrufen?
4. Was müsste ich tun - und was würde dann passieren - wenn ich versuchen wollte die Methode `m` auf der Variable `x` aufzurufen?

5. **Ausnahmen**

   (🇺🇸 *Exceptions*)

   1. Welches ist die Superklasse aller Ausnahmen?
   2. Was ist der Unterschied zwischen *checked* und *unchecked* Ausnahmen?
   3. Wie fange ich eine Ausnahme?
   4. Was muss ich machen, wenn ich eine *checked Exception* nicht fangen will?
   5. Was ist ein `catch` Block.
   6. Warum sollte ich `Error`s nicht fangen?`

Ober- und Unterklassen

1. `class X extends Y { ... }` - Welches ist die Oberklasse und welches die Unterklasse?
2. `class Z extends Y { ... }` - Ist Z eine Unterklasse von X?

Statischer und dynamischer Typ

Gegeben sei folgender Code:

```
class X extends Y { ... }
class Z extends Y { ... }
Y y = new X();
```

1. Welches ist der statische und welches der dynamische Typ von `y`?
2. Kann ich `y` auch mit einem Objekt vom Typ Z initialisieren?
3. Kann ich einer Referenzvariablen `Z z;` ein Objekt vom Typ `X` zuweisen?
4. Kann ich einer Referenzvariablen `Z z;` ein Objekt vom Typ `Y` zuweisen?
5. Wie teste ich, wenn ich eine Referenzvariable `Y y;` habe, ob das Objekt, auf das `y` zeigt, vom Typ `X` ist?
6. Was passiert wenn meine Referenzvariable vom Typ `Y y` mit `null` initialisiert ist, und ich einen Typtest auf `X` durchführe?

Methoden

1. Wann kann ich Methoden in einer Subklasse überschreiben?
2. Was ist der Unterschied zwischen *Method Overloading* und *Method Overriding*?
3. Was ist der Unterschied zwischen einem Konstruktor und einer Methode?
4. Wie kann ich gezielt eine Methode der Superklasse in einer Subklasse aufrufen?
5. Wie kann ich gezielt einen anderen Konstruktor der selben Klasse aufrufen?
6. Welche Methoden hat jede Klasse und warum?

Überschriebene Methoden

Gegeben sei folgender Code:

```
class Y { void p(){println("Y.p");} }
class X extends Y { void p(){println("X.p");} }
class Z extends Y {
   void p(){println("Z.p");}
   void m(){println("Z.m");} }
Y x = new X(); Y z = new Z();
```

1. Was wird ausgegeben bei `x.p();`?
2. Was gibt `x.p();` aus, wenn die Methode `p` in der Klasse `X` nicht überschrieben worden wäre?
3. Wie kann ich die Methode `m` von `Z` auf der Variable `z` aufrufen?
4. Was müsste ich tun - und was würde dann passieren - wenn ich versuchen wollte die Methode `m` auf der Variable `x` aufzurufen?

Ausnahmen

(🇺🇸 *Exceptions*)

1. Welches ist die Superklasse aller Ausnahmen?
2. Was ist der Unterschied zwischen *checked* und *unchecked* Ausnahmen?
3. Wie fange ich eine Ausnahme?
4. Was muss ich machen, wenn ich eine *checked Exception* nicht fangen will?
5. Was ist ein `catch` Block.
6. Warum sollte ich `Error`s nicht fangen?`