



Informatik I

Übungsstunde 10

Herbst 2020

Hausaufgaben

■ Fragen?

Review Bonusaufgaben

Gut gelöst!

Review Bonusaufgaben

Gut gelöst!

- Benutzen von Arrays: limitiert Eingabegrösse.

Review Bonusaufgaben

Gut gelöst!

10'000 100'000 ↖
array[...]

■ Benutzen von Arrays: limitiert Eingabegrösse.
Würde darum Punkteabzug an der Prüfung geben.

~~(ArrayList java.util)~~

Review Hausaufgaben

- Inheritance: Mehrheitlich gut gelöst.
- Encapsulation, visibility modifiers – alle richtig
- Composition or inheritance? – Erster Teil alle richtig, zweiter Teil weniger.

Review Inheritance

Überschreiben von Methoden: Signatur

Get doSomething

Animal.java > ...

```
1 public class Animal {  
2     void doSomething() {  
3         System.out.println("Animal");  
4     }  
5 }
```

protected

Bird.java > ...

```
1 public class Bird extends Animal {  
2     public void doSomething() {  
3         System.out.println("Bird");  
4     }  
5 }
```

@Override

Signatur von doSomething

Signatur von doSomething

Review Inheritance

Welche Methode wird ausgeführt?

➤ Main.java > ...

```
1 public class Main {  
    Run | Debug  
2     public static void main(String[] args) {  
3         Bird bird = new Bird();  
4         bird.doSomething();  
5     }  
6 }
```

➤ Animal.java > ...

```
1 public class Animal {  
2     void doSomething() {  
3         System.out.println("Animal");  
4     }  
5 }
```

➤ Bird.java > ...

```
1 public class Bird extends Animal {  
2     void doSomething() {  
3         System.out.println("Bird");  
4     }  
5 }
```


Review Inheritance

Was referenziert this?

- ↳ (1) Es referenziert die Klasse, in der this aufgerufen wird.
(2) Es referenziert die Instanz, in der this aufgerufen wird.

Bird bird = new Bird();

Bird bird2 = new Bird();

Composition or inheritance?

Code Expert.

Beispiel

Videospiel

mit Skeletten und Zombies.

Videospiel



Videospiel

Du programmierst ein Videospiel. Es gibt zwei HauptFeind: Skelette und Zombies.

Jeder Feind hat eine gewisse Menge von Gesundheit (`int` Gesundheitspunkte).

Skelette können mit einem Schwert verletzt werden, aber nicht mit Feuer.

Zombies können nur mit Feuer verletzt werden.

Schritt 1

Schreibe eine Klasse `abstract class Enemy`, die folgendes unterstützt:

- Speichern der Gesundheit (`int health`);
- Speichern des Namen des Feinds (`String name`);
- Von einem Schwert getroffen werden (dies reduziert die Gesundheit um den Parameter `damage`);
- Informationen über sich selber ausgeben (Name, Gesundheit, ob der Feind noch am leben ist: `health > 0`).

Schritt 1

```
abstract class Enemy {  
    protected int health;  
    private String name;  
  
    public void dealDamage(int damage) {  
        this.health -= damage;  
    }  
  
    public String getInfo() {  
        return this.name + ": " + this.health + "HP, alive: " +  
            (this.health > 0);  
    }  
}
```

Schritt 2

Schreibe einen Konstruktor für `Enemy`, der `health` and `name` initialisiert.

Schritt 2

```
abstract class Enemy {  
    protected int health;  
    private String name;  
  
    protected Enemy(String name, int health) {  
        this.name = name;  
        this.health = health;  
    }  
}
```

Schritt 3

Verfasse eine Klasse `Skeleton`, die die Klasse `Enemy` erweitert (`extends`).

Der Konstruktor sollte einen Parameter entgegennehmen, der dem Level des Feinds entspricht (`int level`).

Der Name dieses Feinds sollte "Level X skeleton" sein; und die Gesundheit sollte `5 * level` sein.

Schritt 3

```
class Skeleton extends Enemy {  
    public Skeleton(int level) {  
        super("Level " + level + " skeleton", level * 5);  
    }  
}
```

Schritt 4

Teste dies in Main.

```
Skeleton skeleton = new Skeleton(2);  
Out.println(skeleton.getInfo());  
skeleton.dealDamage(10);  
Out.println(skeleton.getInfo());
```

Schritt 5

Schreibe eine Klasse `Zombie`, die die Klasse `Enemy` erweitert (`extends`).

Ein `Zombie` sollte nicht anfällig gegen normale Attacken sein.

Er hat 30 Gesundheitpunkte wenn er spawned wird.

Er ist anfällig für Attacken mit der Methode `dealFireDamage`.

Schritt 5

```
class Zombie extends Enemy {  
    public Zombie() {  
        super("Generic zombie", 30);  
    }  
  
    @Override public void dealDamage(int damage) {  
        // no damage  
    }  
  
    public void dealFireDamage(int power) {  
        assert power >= 0 && power <= 10;  
        this.health = this.health * power / 10 - power;  
    }  
}
```

Schritt 6

Teste es in Main aus.

```
Zombie zombie = new Zombie();  
Out.println(zombie.getInfo());  
zombie.dealDamage();  
Out.println(zombie.getInfo());  
zombie.dealFireDamage(5);  
Out.println(zombie.getInfo());
```

Schritt 7

Speichere beide Feind in einem `enemies` Array (unter Verwendung von Polymorphismus).
Iteriere dann über den Array und rufe `getInfo` bei jedem Feind auf.

Schritt 7

```
Enemy enemies[] = new Enemy[2];  
enemies[0] = zombie;  
enemies[1] = skeleton;  
  
for (int i = 0; i < enemies.length; i++) {  
    Out.println(enemies[i].getInfo());  
}
```

Nächste Aufgaben

- Polymorphic Animals
- Polymorphism and Overrides
- Mr Brush
- Comparing Relational Numbers