ETH zürich



Informatik I

Übungsstunde 10

Herbst 2020

Hausaufgaben

■ Fragen?

Review Bonusaufgaben

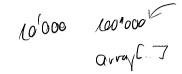
Gut gelöst!

Review Bonusaufgaben

Gut gelöst!

■ Benutzen von Arrays: limitert Eingabegrösse.

Review Bonusaufgaben



Gut gelöst!

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



Review Hausaufgaben

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

Review Inheritance

Got dosullia Überschreiben von Methoden: Signatur protectel Animal.java > ... Bird.java > ... public class Animal public class Bird extends Animal { (void doSomething()) { public void doSomething() { System.out.println("Animal"); System.out.println("Bird"); 6 Overricle Shudh vou do Someth's Shudu vou do Someth's

Review Inheritance

Welche Methode wird ausgeführt?

```
Main.java > ...

public class Main {
    Run | Debug

| public static void main(String[] args) {
    Bird bird = new Bird();
    bird.doSomething();
}
```

```
Animal.java > ...
    public class Animal {
       void doSomething() {
         System.out.println("Animal");
  5
Bird.iava > ...
 1 public class Bird extends Animal {
     void doSomething() {
        System.out.println("Bird");
 4
 5
```

Review Inheritance

Was referenziert this?

7

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.

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).

```
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);
```

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

```
abstract class Enemy {
    protected int health;
    private String name;

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

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.

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

Teste dies in Main.

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

Out.println(skeleton.getInfo());

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.

```
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;
```

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());
```

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

```
Enemy enemies[] = new Enemy[2];
enemies[0] = zombie;
enemies[1] = skeleton;

for (int i = 0; i < enemies.length; i++) {
    Out.println(enemies[i].getInfo());
}</pre>
```

Nächste Aufgaben

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