LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Fakultät für Betriebswirtschaft
Munich School of Management

Basics in Programming for MMT

Session 7 – Parents and Children

MMT

www.mmt.bwl.lmu.de

Scope of the Session

1. Repetition

- Class

2. Theory

- Can we further simplify our Code?
- Inheritance
- Super
- Type
- Calling Parent Methods

3. Next

- Session 8

4. Tutorial

- PONG

Repetition

Repetition

Class (1/2)

- A `class` allows us to define data structures.
- The class (`Ball`) is the abstract description and the objects (`b`) are instances of that class.
- Each object contains its own set of variables defined in the class as fields.

```
Ball b;

void setup () {
  size(600,600);
  b = new Ball (235,237,52);
}

void draw () {...}

class Ball {
  float x;
  float y;
  float d;

  Ball (float x, float y, float d) {
    this.x = x;
    this.y = y;
    this.d = d;
    }
}
```

Repetition

Class (2/2)

Keyword `class` + class name

```
class Ball {
```

Fields

```
  float x;
  float y;
  float d;
```

Constructor: class name + arguments

```
  public Ball (int x, int y, int d) {
    this.x = x;
    this.y = y;
    this.d = d;
  }
```

Methods

```
  void move () {
    x = x+1;
    y = y+1;
  }
```

End of class

```
}
```

Theory

Theory

Can we further simplify our Code?

- Sometimes different classes share properties and functionalities.
- Can they share their similarities?

```
class Ball {
  float x;
  float y;
  float dx;
  float dy;
  //...
  void move () {...}
  void plot () {...}
}
class Bat {
  float x;
  float y;
  //float dx;
  float dy;
  //...
  void move () {...}
  void plot () {...}
}
```

Theory

Inheritance (1/2)

- Java allows us to define child parent relationships between classes.
- A child inherits all **fields** and **methods** from the parent class.
- `extends` marks a class as the child. After `extends` the name of the parent class follows.

```
class A {
  float x;
  ...
  void doSomething () {...}
}

class B extends A {
  // float x;
  ...
  // void doSomething () {...}
}

A a;
B b;

void setup () {
  a = new A();
  b = new B();
  a.doSomething();
  b.doSomething();
}
```

Theory

Inheritance (2/2)

- We can outsource the properties that classes have in common to a parent class.
- These properties do not have to be declared later in the individual classes because they are inherited.

```
class Element {
  float x;
  float y;
  float dx;
  float dy;
  Element (float x, float y, float dx,
    this.x = x;
    this.y = y;
    this.dx = dx;
    this.dy = dy;
  }
  void move () {
  x = x+dx;
  y = y+dy;
  }
}

class Ball extends Element {
  ...
}
class Bat extends Element {
  ...
}
```

Theory

Super (1/2)

- In the constructor of a child class, we have to call the `super` constructor.
- This refers to the constructor defined in the parent class.
- By calling we initialize the parent part of the object.

```
class Ball extends Element {
  // x,y,dx,dy from Element
  float d;

  Ball (float x, float y, float d) {
    super(x,y,5,5); // from Element
    this.d = d;
  }

  ...
}
```

Theory

Super (2/2)

- A class can have **multiple** children.
- But only **one** parent?

```
class A {...}

class B extends A {...}

class C extends B {...}

class D extends B {...}
```

Theory

Type

- Objects of the shown classes **can** all be stored in variables of type `A`.
- But if we call them, we can just use the properties of the defined datatype.

```
class A {...}

class B extends A {...}

class C extends B {...}

class D extends B {...}

A a1 = new A();
A a2 = new B();
A a3 = new C();
A a4 = new D();
```

Theory

Calling Parent Methods (1/3)

- If we call a method defined in a parent class on a child object, the method defined in the parent class is executed.

```
class Element {
  ...
  void move () {
  x = x+1;
  y = y+1;
  }
}

class Ball extends Element {
  ...
}

Ball b;

...

void draw () {
  ...
  b.move(); // -> speed = 1
}
```

Theory

Calling Parent Methods (2/3)

- We can overwrite methods in child classes.
- If a method is called on a child object and the method is defined in both, parent and child classes, the child class method is executed.

```
class Element {
  ...
  void move () {...}
}

class Ball extends Element {
  ...
  void move () {
    x = x+8;
    y = y+8;
}

Ball b;
...

void draw () {
  ...
  b.move(); // -> speed = 8
}
```

Theory

Calling Parent Methods (3/3)

- If a method is defined in both, parent and child classes, but we want to refer to the method of the parent class, we have to use `super` to specify the method.

```
class Element {
  ...
  void move () {...}
}

class Ball extends Element {
  ...
  void move () {
    super.move();
    println("extra code executed");
}

Ball b;
...

void draw () {
  ...
  b.move(); // -> speed = 1
}
```

Next

Next

Session 8

- Interfaces
- Classes sharing functionalities on a meta level

Tutorial

Tutorial

PONG

1. Write a child class of Ball that has a different movement pattern (overwrite: move()): some jitter (randomness) in addition to the normal movement
2. If the Ball leaves the window decide randomly to overwrite the Ball (reinitialize the variable = calling constructor) object either with the classic Ball or your new defined RandomBall