LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Fakultät für Betriebswirtschaft
Munich School of Management

Basics in Programming for MMT

Session 5 – Classes and Objects

MMT

www.mmt.bwl.lmu.de

Scope of the Session

1. Repetition

- Functions
- KeyPressed() and KeyReleased()

2. Theory

- Classes
- Defining Classes
- Class
- This & Objects

3. Next

- Session 6

4. Tutorial

- Reference
- PONG

Repetition

Repetition

Functions

- Functions allow us to use **generalized** sets of instructions.
- A set of commands is performed when calling the function, using the **attributes** from the attribute list.
- Functions can return values of datatypes or do not return any value (`void`).

```
void printRandomCharacters () {
    for (int i=0; i<100; i = i+1) {
          print(char((int)random(255)));
    }
}

int double (int v) {
    return 2*v;
}
```

Repetition

KeyPressed() and KeyReleased()

- If a key is pressed/released `keyPressed()`/`keyReleased()` is executed.
- `key` contains the value of the last pressed key.

```
void setup () {}

void draw () {}

void keyPressed () {
    if (key == 'w') {}
    if (key == 's') {}
}

void keyReleased () {}
```

Theory

Theory

Classes (1/2)

- We already defined a “ball” based on multiple variables.
- If we want to handle multiple balls, we have to either duplicate code or restructure the code to run based on arrays.
- Is there a way of teaching the computer what we understand by this?

```
float x,y,dx,dy,d;

void setup () {
  size(600,600);
  x = 0;
  y = 0;
  dx = 1;
  dy = 1;
  d = 50;
}

void draw () {
  background(0);

  x = x+dx;
  y = y+dy;

  fill(255,0,0);
  ellipse(x,y,d,d);
}
```

Theory

Classes (2/2)

- By teaching the computer we can define an abstract description of a ball.
- Such descriptions are called classes and contain data as well as functionalities.
- We can also use multiple instances.

```
Ball b;

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

void draw () {
  background(0);

  b.move();
  b.checkborders();
  b.draw();
}
```

Theory

Defining Classes

- We can add class definitions outside of `setup()` and `draw()`.
- A class consists of fields, constructors.

```
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;
  }
}
```

Theory

Class (1/4)

Keyword `class` + class name	`class Ball {`
Fields	`float x;` `float y;` `float d;`
Constructor: class name + arguments	see below
End of class	`}`

```
class Ball {
```

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

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

```
}
```

Theory

Class (2/4)

- Right now, the class is just a `container` for multiple variables.
- We can access them by using the `.` operator.

```
Ball b;

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

void draw () {
  b.x = b.x+1;
  b.y = b.y+1;
}

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

  ...
}
```

Theory

Class (3/4)

- But a class can also contain functionalities that can be called using the `.` operator on the object.
- A class it the **abstract description** of a data structure and related functionalities.
- Objects are **instances** of classes.

```
Ball b;

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

void draw () {
  b.move();
}

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

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

Theory

Class (4/4)

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

This & Objects (1/2)

- Inside the constructor we use `this` to prevent naming conflicts.
- A field of the class as well as the passed argument are both called `x`.
- `this` refers to the field of the current object we are working on.

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

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

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

Theory

This & Objects (2/2)

- A `class` itself is just the description of the data structure.
- The object is the created instance.
- Each object has its own set of variables.
- Methods are called on objects. Variables used in the method refer to the variables of the object we are calling the method on.

```
Ball b;
Ball c;

void setup () {...}

void draw () {
  background(0);
  b.move();
  c.move();
}

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

  ...

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

Next

Next

Session 6

- Using objects
- Passing objects vs. passing variables

Tutorial

Tutorial

Reference

1. Go to: processing.org/reference
2. Check out PVectors and think of how to use them to describe the position and direction of our moving objects (bats, balls)

Tutorial

PONG

1. Expand the example code of yesterday’s lecture (moving bats).
   This time the ball should bounce off the bats and pass by.
2. Reset the ball if it leaves the window.