# Introduction to Programming

## Lecture 5: inheritance

Ben Ruijl

Nikhef Amsterdam and Leiden University

November 8, 2016

# Printing class

Say you want to print a class to the console:

```
1  class Vector {
2      float x, y;
3      Vector(float x, float y) {
4          this.x = x;
5          this.y = y;
6      }
7  }
8
9  Vector a = new Vector(200, 300);
10 println(a);
```

```
sketch_141012a$1Vector@7931a5af
```

# Printing class

- The default string representation is the location in memory
- To change this, we have to override the `toString()` function:

```
1 class Vector {
2   @Override
3   String toString() {
4      return "(" + x + "," + y + ")";
5   }
6 }
7
8 Vector a = new Vector(200, 300);
9 println(a);
```

```
(200.0,300.0)
```

# Comparing classes

- We have to be careful with comparing classes
- == compares the memory addresses

```
1 void setup() {
2   String a = "HI";
3   String b = "hi".toUpperCase();
4
5   println(a == b);
6 }
```

```
false
```

# Comparing classes

- We should use `equals`:

```
1 void setup() {
2   String a = "HI";
3   String b = "hi".toUpperCase();
4
5   println(a.equals(b));
6 }
```

```
true
```

# Comparing classes

- For custom classes we should override the default `equals`
- You should compare `Vector` to any `Object`

```
1  class Vector {
2    @Override
3    boolean equals(Object o) {
4       if (!(o instanceof Vector)) return false;
5       Vector b = (Vector)o; // cast to Vector
6       return x == b.x && y == b.y;
7    }
8  }
9  Vector a = new Vector(2, 3), b = new Vector(2, 3);
10 println(a == b);
11 println(a.equals(b));
```

# Inheritance

- Every class is derived from the base class `Object`
- We can override functions from the base class: `toString`, `equals`, etc.
- We could also build our own class and let other classes be derived from it
- This is called inheritance

# A base class for cars

Say we write a class for a car:

```
1 class Car {
2     color colour;
3
4     void start() { }
5     void accelerate() { }
6     void break() { }
7     String toString() { return "Car"; }
8 }
```

# Inheritance

- We want to distinguish between family cars and race cars
- Race cars have a turbo feature
- Do we have to copy the Car class?
- We can extend a Car to form a RaceCar:

```
1 class RaceCar extends Car {
2     void turbo() { println("Doing turbo!") }
3     @Override
4     void accelerate() { turbo(); }
5     @Override
6     String toString() { return "Race car!"; }
7 }
```

- All functions and class variables are inherited

# Example

We have overridden `accelerate()` and `toString()`:

```
1 Car familyCar = new Car();
2 familyCar.accelerate();
3 println(familyCar);
4 RaceCar raceCar = new raceCar();
5 raceCar.accelerate();
6 println(raceCar);
```

```
Car
Doing turbo!
Race car!
```

# Abstraction

Since `RaceCar` is a `Car`, we could also say:

```
1 Car raceCar = new RaceCar();
2 println(raceCar);
```

```
Race car!
```

**Abstraction**

It still 'knows' it is a `RaceCar` even though the type is now `Car`!

# Arrays of derived classes

We can treat all cars the same, but different things happen:

```
1 Car[] cars = new Cars[3];
2 cars[0] = new Car();
3 cars[1] = new RaceCar();
4 cars[2] = new Car();
5
6 for (int i = 0; i < cars.length; i++) {
7     println(cars[i]);
8 }
```

```
Car
Race car!
Car
```

# Example in games

For games this may be useful:

```
1  class GameObject {
2      int x, y;
3      color fillColour;
4
5      void draw() { };
6  }
7
8  List<GameObject> objects = new ArrayList<GameObject>();
9
10 for (GameObject e : objects) {
11     e.draw();
12 }
```

# Constructors

If the base class has a constructor, call it with **super**:

```
1 class Base {
2   Base(int x) { };
3 }
4
5 class Derived extends Base {
6   Derived() {
7     super(10);
8   }
9 }
```

# Tabs

Some tips:

- For your game, you are going to write quite some code
- Make tabs to split code into multiple files
- If you have large classes, put them in seperate files

# Writing text

```
1  PFont f;
2
3  void setup() {
4    size(640,640);
5    f = createFont("Arial",40,true);
6    textFont(f,40);
7    fill(0); // text colour
8  }
9
10 void draw() {
11   textAlign(CENTER);
12   text("Hello world!",320,320);
13 }
```