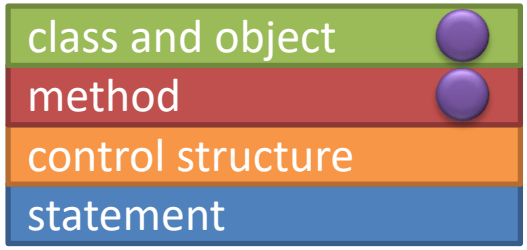
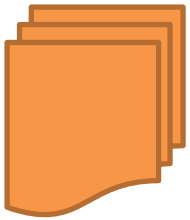
Custom Data Types: Creating Your Own Classes





11 Creating Your Own Data Types

Dr James Montgomery, james.montgomery@utas.edu.au

Classes and Objects

A class

Rectangle

```
int w;
int h;
int area()
static int count()
```

"Blueprint" for objects

Container for static (class-level) methods

Objects of that class

```
w == 4;
h == 4;
area()
```

```
w == 2;
h == 6;
area()
```

```
w == 7;
h == 4;
area()
```

Have *state* (the value of their properties)

Have behaviour (methods)

Object variable is a *reference* to the object's location in memory

Objects must be instantiated with **new**

```
Rectangle r = new Rectangle();
```

Class declaration

```
public class ClassName {
    data (variable) declarations
    method declarations
}
```

Class declaration

(very basic) example:

```
public class Rectangle {
    private int width, height;

    public int area() { … }

    public static int count() { … }
}
```

*Defines the **data** (width & height) and **methods** (area) of all objects of class Rectangle*

This example code must be stored in the file Rectangle.java

Method declaration

Object method

```
/** comments */
access return type identifier ( parameter list ) {
    local variable declarations
    code to do the work
    return statement (if return type is not void)
}
```

(overly complicated) example:

```
/** Returns the area of the rectangle. */
public int area() {
    int a;
    a = width * height;
    return a;
}
```

Common components

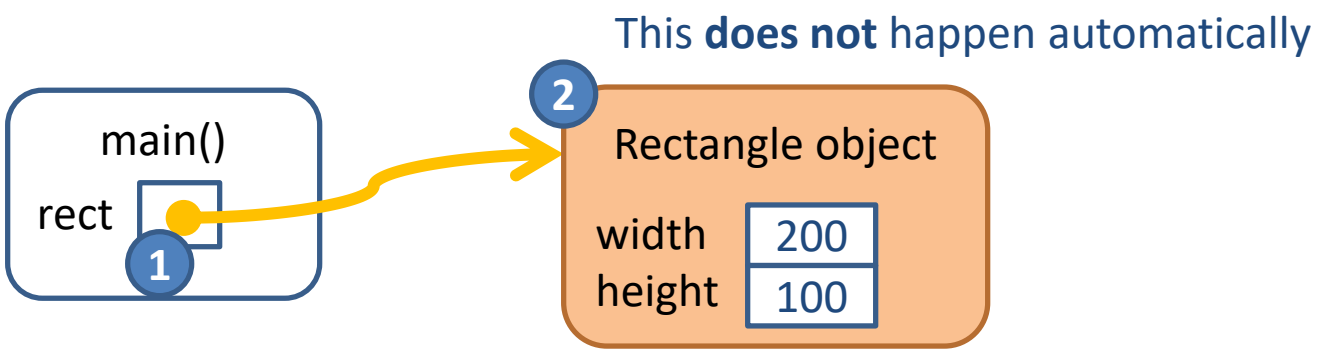
Suggested class declaration

```
public class ClassName {
    data (variable) declarations
    constructor
    getters & setters
    public String toString() {
        statement(s)
    }
}
```

Method declarations

Preparing an object for use

1. Rectangle rect;
2. rect = new Rectangle(); The **new** operator calls the constructor



A *constructor* is a special kind of method called when an object is first created

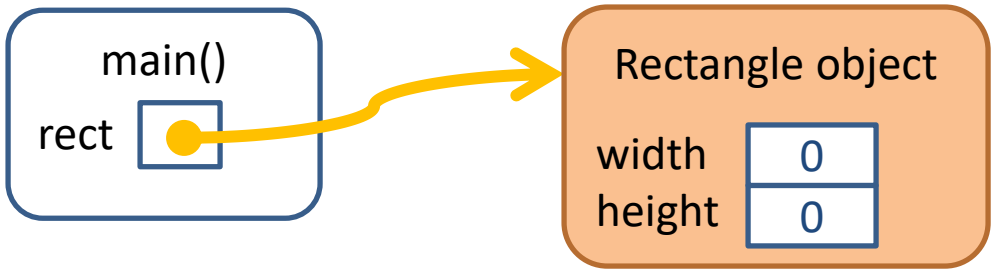
It contains code to set up the object with sensible initial values for its data fields

Rectangle with no constructor

```
Rectangle rect;
rect = new Rectangle();
```

If no constructor specified a default one is created (behind the scenes)

Instance data are set to defaults: numbers to zero, booleans to **false**, objects to **null**



Template for a constructor

```
access ClassName ( parameter list ) {
    statements
}
```

Constructor

Example:

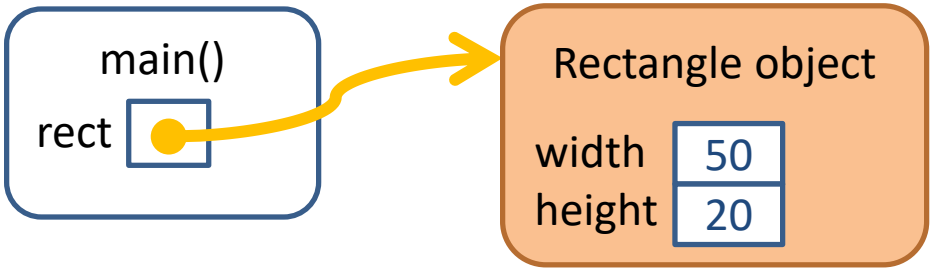
```
public Rectangle(int w, int h) {
    width = w;
    height = h;
}
```

Constructor implementation

```
Rectangle rect;
rect = new Rectangle(50, 20);
```

Rectangle.java

```
public class Rectangle {
   …
   public Rectangle(int w, int h) {
      width = w;
      height  = h;
   }
   …
}
```



Multiple alternative constructors can be specified

Getters and setters

‘Getters’ and ‘setters’ are methods for providing access to the data inside an object

Basic getter & setter

```
public type get PropertyName() {
    return propertyName;
}
public void set PropertyName(type paramName) {
    propertyName = paramName;
}
```

Getters and setters

Rectangle.java

```
public class Rectangle {
   private int width;
   private int height;
   …
   public int getWidth() {
      return width;
   }
   public void setWidth(int w) {
      width = w;
   }
   public int getHeight() {
      return height;
   }
   public void setHeight(int h) {
      height = h;
   }
}
```

Comments omitted to fit on slide

Although this *looks* like a waste of time and effort (why not just make width and height public?) the contents of setWidth() and setHeight() could be more complex

For example, we could prevent width from being negative:

```
public void setWidth(int w) {
   if (w >= 0) {
      width = w;
   }
}
```

toString

Every kind of object has a method called toString() which returns a String representation of the object

It is called automatically when a String is needed, such as when calling println()

The default implementation is only useful for limited debugging:

```
Rectangle r = new Rectangle(10, 20);

System.out.println(r);
```

produces

```
Rectangle@12bc500
```

This is the object's memory address in hexadecimal
(not at all user friendly)

You can and should provide your own implementation

Rectangle.java

```
public class Rectangle {
   private int width;
   private int height;

   …

   public String toString () {
      return width + "x" + height
                          + " rectangle";
   }
}
```

now

```
System.out.println(r);
```

produces

```
10x10 rectangle
```

Implementing *any* class