

1B1b

Classes in Java

Part I

1 Copyright © 2004, Graham Roberts

Department of Computer Science



Agenda

- Defining simple classes.
- Instance variables and methods.
- Objects.
- Object references.

2 Copyright © 2004, Graham Roberts

Department of Computer Science



Reading

- You should be reading:
 - Part I chapters 6,9,10
- And browsing:
 - Part IV chapter 30

3 Copyright © 2004, Graham Roberts

Department of Computer Science



Classes & Objects

- We want to design programs in terms of classes and objects.
- Write the classes using the programming language.
- Use the objects when the program runs.

4 Copyright © 2004, Graham Roberts

Department of Computer Science



Classes

- Reminder – a class consists of:
 - A collection of *instance methods*.
 - A collection of *instance variables* to represent the *state* of an object (instance of the class).

5 Copyright © 2004, Graham Roberts

Department of Computer Science



A very simple example

```
class MyClass
{
    private int code = 10;
    private String name = "Widget";
    public int getCode()
    { return code; }
    public String getName()
    { return name; }
    public void setCode(int n)
    { code = n; }
    public void setName(String s)
    { name = s; }
}
```

2 instance variables,
4 methods

6 Copyright © 2004, Graham Roberts

Department of Computer Science



Class MyClass

- By convention a class name always starts with a capital letter.
- *Instance objects* can be created using:

```
MyClass myObj = new MyClass() ;
```
- As many objects as you need can be created.

7

Copyright © 2004, Graham Roberts

Department of Computer Science



A MyClass Object

- Has two instance variables.
- And two methods, as declared by its class.

myObj : MyClass
code = 10 name = "Hello"

8

Copyright © 2004, Graham Roberts

Department of Computer Science



Objects

- Each object has its own private set of variables.
- Each object is independent.

: MyClass
code = 3 name = "Cog"

: MyClass
code = 10 name = "Widget"

: MyClass
code = 42 name = "Wheel"

9

Copyright © 2004, Graham Roberts

Department of Computer Science



Private

- A class defines a scope.
- Declaring a method or variable *private* means that it can only be accessed within the scope of the class.
 - This means within a method body or an instance variable initialisation expression.

10

Copyright © 2004, Graham Roberts

Department of Computer Science



Public

- Methods and variables declared public can be accessed by anything that has a *reference* to an object of the class.
- They form the public interface of objects of the class.
 - The services the object can perform.

11

Copyright © 2004, Graham Roberts

Department of Computer Science



Encapsulation

- Public and private are how encapsulation is enforced.
- Good practice states:
 - Instance variables are *always* private.
 - Only a *minimal* number of methods should be made public.

12

Copyright © 2004, Graham Roberts

Department of Computer Science



Questions?

13 Copyright © 2004, Graham Roberts

Department of Computer Science



Type MyClass

- MyClass is actually a new *type*.
- This is why declarations such as:

```
MyClass myObj = new MyClass();
```

are possible.
- In fact, MyClass is a *User Defined Type*.

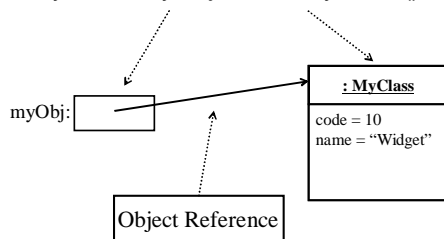
14 Copyright © 2004, Graham Roberts

Department of Computer Science



Object references

```
MyClass myObj = new MyClass();
```



15 Copyright © 2004, Graham Roberts

Department of Computer Science



References

- A variable of a class type holds a *reference* to an object.
- It doesn't hold the object itself.
- The variable can go out of scope but the object can still exist (providing it is referenced by some other variable).
- One object can be referenced by many references and, hence, variables.

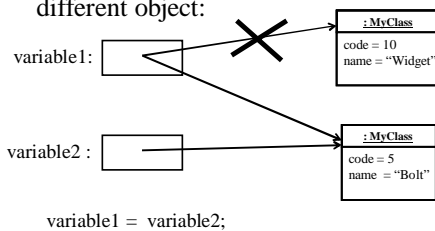
16 Copyright © 2004, Graham Roberts

Department of Computer Science



Class Type Assignment

- Assignment means storing a reference to a different object:



17 Copyright © 2004, Graham Roberts

Department of Computer Science



So,

- When you declare a variable of class type, you get a container that holds an object reference.
- Assigning an “object to a variable” means storing a reference to the object in the variable.


18 Copyright © 2004, Graham Roberts

Department of Computer Science



Null Reference

- null
- No object is referenced, so no methods can be called.
- `MyClass obj = null;`
- Default value if variable not initialised.

variable3 : 

19 Copyright © 2004, Graham Roberts

Department of Computer Science



Questions?

20 Copyright © 2004, Graham Roberts

Department of Computer Science



Calling methods

- Given an object reference, a method can be called:


```
MyClass myObj = new MyClass();
int n = myObj.getCode();
String s = myObj.getName();
```
- Only methods declared* by class `MyClass` can be called.

* Actually inherited methods can also be called. We will find out about this later in the course.

21 Copyright © 2004, Graham Roberts

Department of Computer Science



NullPointerException

- Caused by calling method on a null reference.
- No object, so no method can be called.
- If the error occurs find out why the variable is not referencing an object.

22 Copyright © 2004, Graham Roberts

Department of Computer Science



Method parameters

- We know that methods can have parameters and return a result.
- For example:


```
public int f(int n)
{
    return n*2;
}
```

23 Copyright © 2004, Graham Roberts

Department of Computer Science



Object Reference Parameters

- You can pass an object reference as a parameter to a method:


```
void someMethod(MyClass myObj)
{
    ...
}
```
- A parameter variable is declared as normal.

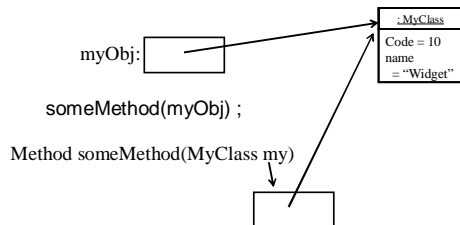
24 Copyright © 2004, Graham Roberts

Department of Computer Science



Parameters & References

- But, consider what happens:



25 Copyright © 2004, Graham Roberts

Department of Computer Science



Object Parameters

- The parameter value is an *object reference*, not an object.
- The parameter variable is initialised to hold a *copy* of the reference.
- The object is *not* copied.

26 Copyright © 2004, Graham Roberts

Department of Computer Science



Consequences

- If an object reference is passed as a parameter then:
 - Changing the object inside the method changes the object outside the method.
 - They are the same object!
- Don't forget that arrays are objects.
 - Changing elements in an array parameter object changes the array outside the method.

27 Copyright © 2004, Graham Roberts

Department of Computer Science



Remember Primitive Types?

- Values of primitive types (int, char, long, boolean, etc.) are stored directly in variables using a binary representation.
- They are *not* objects.
- You can't have references to values of primitive types.
 - But there are classes to represent values of primitive types.
 - Class Integer, Double, Character, etc.

28 Copyright © 2004, Graham Roberts

Department of Computer Science



Primitive type parameters

- When a value of a primitive type is passed as a parameter, it is *always* copied.
- The parameter variable is initialised to a *copy* of the argument value.

29 Copyright © 2004, Graham Roberts

Department of Computer Science



Call-by-value

- The parameter passing mechanism used by Java is called "Call-by-value".
- *This means that the value of a parameter is always copied and a parameter variable initialised with the copy.*
- Objects are not passed as parameters, only references.

30 Copyright © 2004, Graham Roberts

Department of Computer Science



Parameter passing reminder

- A *value* is computed in a method call:
`obj.f(a + b);`
- The value is used to initialise the parameter variable of the called method:
`public void f(int n) { ... }`

31

Copyright © 2004, Graham Roberts

Department of Computer Science



“Variable is passed...”

- Beware, this means the value held in the variable is passed as a parameter:
`obj.f(x);`
- The variable itself is not passed.
- The value of the variable is not changed by the method called.
 - But an object referenced by the variable can change.

32

Copyright © 2004, Graham Roberts

Department of Computer Science



Summary

- Seen how to construct very simple classes.
- Methods and instance variables.
- Object references.
- References and parameter passing.
- Call-by-value.

33

Copyright © 2004, Graham Roberts

Department of Computer Science

