

1B1b

Classes in Java

Part II

1 Copyright © 2004, Graham Roberts

Department of Computer Science



Creating classes

- Suppose you want to write a program to store information about your collection of books...
- (Very exciting I know but we have to start somewhere!)
- What classes might you need?

2 Copyright © 2004, Graham Roberts

Department of Computer Science



Two classes

- One class can provide objects to represent books.
 - Class Book.
- A second class can provide an object to represent the collection of books.
 - Class Library.
 - Allowing books to be added, removed and searched for.

3 Copyright © 2004, Graham Roberts

Department of Computer Science



A Book Class

- What information does a book object need to store?
 - Title
 - Author(s)
 - ISBN
 - Publisher
 - Publication date
- Should all be relevant to context we use the class in.

4 Copyright © 2004, Graham Roberts

Department of Computer Science



Behaviour

- What operations should a book object have?
 - It needs to be initialised when created.
 - Provide access to the book information.
- Once created does it need to be changed?
 - No, probably not.
 - If a change is needed a new object can be created.

5 Copyright © 2004, Graham Roberts

Department of Computer Science



Book.1

```
public class Book
{
    private String title = ??? ;
    private String author = ??? ;
    // etc.
    public String getTitle() { return title; }
    public String getAuthor() { return author; }
    // etc.
}
```

6 Copyright © 2004, Graham Roberts

Department of Computer Science



Initialisation

- String title = *what?*
- The instance variables need to be initialised differently for each book.
- How is this done if no methods for changing the object state are provided?

7

Copyright © 2004, Graham Roberts

Department of Computer Science



Creation + Initialisation

- What we need is a way of combining object creation with the required initialisation.
- Fortunately, Java provides us with a proper way of doing this.

8

Copyright © 2004, Graham Roberts

Department of Computer Science



Constructor

```
Book book = new Book( ... );
```

- When an object is created using new, a *constructor method* is called.
- It can initialise the object.

9

Copyright © 2004, Graham Roberts

Department of Computer Science



Constructor Method

- A method but subject to special rules:

<pre>public Book(...) { title = ... ; author = ... ; // etc. }</pre>	<ul style="list-style-type: none"> • Same name as class name. • Cannot return a value. • Cannot be called like a normal method.
--	--

10

Copyright © 2004, Graham Roberts

Department of Computer Science



Constructor Arguments

- Constructors can take arguments.
 - The arguments give the values needed to initialise the object.
- ```
public Book(String aTitle, String anAuthor, etc...)
{
 title = aTitle;
 author = anAuthor;
 // and so on
}
```

11

Copyright © 2004, Graham Roberts

Department of Computer Science



## Providing the arguments

- ```
Book book = new Book("MyBook", "Me", etc...);
```
- The parameters to the constructor are given in the parameter list in the new expression.

12

Copyright © 2004, Graham Roberts

Department of Computer Science



New - what happens?

- The new operator causes two things to happen.
 1. An object with the correct set of instance variables is created
 2. The constructor is called with the given arguments.
 - Then the reference to the new initialised object is returned.

13 Copyright © 2004, Graham Roberts

Department of Computer Science



The Guarantee

- The language rules guarantee that the constructor will be called when a Book object is created using new.
- You cannot prevent this happening.
- Object must be initialised or else!
- (And you cannot create objects any other way.)

14 Copyright © 2004, Graham Roberts

Department of Computer Science



No constructor?

- Will the guarantee hold if you don't provide a constructor?
- Yes – as the compiler will provide one for you!!
- It doesn't do anything except act as a place holder but it is there.
 - Instance variables are initialised to default values.

15 Copyright © 2004, Graham Roberts

Department of Computer Science



Parameter Variable Naming Idiom

```
public Book(String title, String author, etc...)
{
    this.title = title;
    this.author = author;
    // and so on
}
```

- Parameter and instance variables have same names, so,
- Use this.name to refer to instance variable
 - Avoids need to think up new names,
 - But possibly more open to mistakes/confusion?

16 Copyright © 2004, Graham Roberts

Department of Computer Science



Questions?

17 Copyright © 2004, Graham Roberts

Department of Computer Science



Destructing

- What happens to an object when it is no longer needed – it has no references?
- It might, at some point, be destroyed by the *garbage collector* and its memory reused.
- Can we get any control over this?
- Yes (sort of), but normally we don't need to and for now we won't try.
 - And the mechanism is unpredictable.
 - There is no guarantee an object will be removed from memory while a programming is running.

18 Copyright © 2004, Graham Roberts

Department of Computer Science



Remember...

```
public class Book
{
    private String title = ? ;
    private String author = ? ;
```

- The instance variables don't have to be initialised where they are declared.
- However, they must be assigned a value before they can be used in any other way (which is what the constructor does).

19 Copyright © 2004, Graham Roberts

Department of Computer Science



But good practice...

```
public class Book
{
    private String title = "Invalid";
    private String author = "Nobody";
```

- Always initialise variables directly just in case!
- Here the values will simply be overwritten but we might make a mistake and forget.
- Choose defaults that make you notice.

20 Copyright © 2004, Graham Roberts

Department of Computer Science



But second thoughts...

```
public class Book
{
    private String title;
    private String author;
```

- Do we really want unnecessary initialisations?
- Always use constructor?
- Weigh up the options and make a choice.

21 Copyright © 2004, Graham Roberts

Department of Computer Science



So,

- We can write a Book class.
 - With a set of instance variables to allow an object to represent a particular book.
 - A set of methods to access information about a book.
 - A constructor to initialise a book.

22 Copyright © 2004, Graham Roberts

Department of Computer Science



Classes and .java files

- We are going to have two classes - one file or two? File names?
- Each class should be put in a *separate* .java file.
- A file must be named after the class it holds.

23 Copyright © 2004, Graham Roberts

Department of Computer Science



Book.java & Library.java

- Create two files with these names.
- Each file can be compiled separately.
- Two .class files will be created.

24 Copyright © 2004, Graham Roberts

Department of Computer Science



Lots of files

- A program with lots of classes will have lots of .java files.
- We don't want everything crammed into one file.
- Keep files (and classes) to a manageable size!

25

Copyright © 2004, Graham Roberts

Department of Computer Science



Questions?

26

Copyright © 2004, Graham Roberts

Department of Computer Science



Class Library

- It will need to store a collection of Book objects.
- An array could be used:
`private Book[] books = new Book[???];`
- How big is the array?
- Estimate, by considering the size of a typical collection of books?

27

Copyright © 2004, Graham Roberts

Department of Computer Science



More Flexibility

- Better to use an ArrayList:
`private ArrayList books = new ArrayList();`
- No longer have to worry about number of books.
 - Deferred size decisions.
 - What if library is very large?

28

Copyright © 2004, Graham Roberts

Department of Computer Science



Library public methods

- `public void addBook(Book aBook);`
- `public Book searchForTitle(String aTitle);`
- `public void removeBook(String aTitle);`
- and others?
 - Select those which correspond to the needs of the program you are trying to write.

29

Copyright © 2004, Graham Roberts

Department of Computer Science



Adding a book

- The details of the book need to be input.
- A new Book object created.
- And the addBook method called.
- But, what if library is full?
 - What about duplicates?
- Lots of possible issues – resolve by referring to requirements or talking to client.

30

Copyright © 2004, Graham Roberts

Department of Computer Science



Finding a book

- public Book searchForTitle(String aTitle);
- Given the title string we can search through the ArrayList and compare title strings.
- But...

31 Copyright © 2004, Graham Roberts

Department of Computer Science



Book...

- public Book searchForTitle(String aTitle);
- This methods returns a book object reference.
- A reference to an object stored privately by the Library object.
- Good idea?

32 Copyright © 2004, Graham Roberts

Department of Computer Science



Well...

- In this case it doesn't matter as the value of a Book object cannot be changed.
- However, in general this needs to be considered carefully.
- Return a reference to a stored object or a reference to a *copy* of a stored object?

33 Copyright © 2004, Graham Roberts

Department of Computer Science



No book...

- Wait, what if we search for a book not in the library?
 - public Book searchForTitle(...)
- Or library is empty?
- What should be returned?

34 Copyright © 2004, Graham Roberts

Department of Computer Science



Return value?

- Nothing?
- Book at index 0?
- Random book?
- "Default" book?
- Some sort of marker?

35 Copyright © 2004, Graham Roberts

Department of Computer Science



Default book

- Could return default ("Invalid" by Nobody).
- But client code (code that calls search method) must check title.
- What if there is a real book with title "Invalid"?
 - Choose a string unlikely to ever be a title?
 - "skljdhfsdkjfhdkjsjfh"

36 Copyright © 2004, Graham Roberts

Department of Computer Science



null

- Really want to return a value that can never be confused with a valid book object.
- Use the value null.
 - Null Reference.

37 Copyright © 2004, Graham Roberts

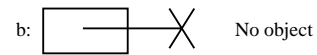
Department of Computer Science



null book

- When a variable of class Book holds the value null, it does not reference an object.

```
Book b = null;
```



38 Copyright © 2004, Graham Roberts

Department of Computer Science



null (3)

- Methods cannot be called on a null reference.
 - There is no object.

```
Book b = null;
b.getTitle(); // Error
```

```
java.lang.NullPointerException
at Book.main(Book.java:12)
Exception in thread "main"
Process Book exited abnormally with code 1
```

39 Copyright © 2004, Graham Roberts

Department of Computer Science



So...

```
public Book searchForTitle(String aTitle)
{
    for (int i = 0 ; i < books.size() ; i++)
    {
        Book book = (Book)books.get(i);
        if (book.getTitle().equals(aTitle))
        {
            return book;
        }
    }
    return null; // Book not in library.
}
```

40 Copyright © 2004, Graham Roberts

Department of Computer Science



But...

- The client code, may get a null reference:


```
Book b = myLibrary.searchForTitle(aTitle);
if (b != null)
{
    // Do something with the book
}

```
- If the test is missing, risk a NullPointerException
 - Nothing is free!

41 Copyright © 2004, Graham Roberts

Department of Computer Science



null, ==, !=

- null can be compared to another reference using == or !=.


```
if (obj1 != null) ...
```
- Two valid references can be compared:


```
if (obj1 == obj2) ...
```
- True if both refer to the same object.

42 Copyright © 2004, Graham Roberts

Department of Computer Science



Questions?

43 Copyright © 2004, Graham Roberts

Department of Computer Science



Library constructor

- Doesn't need any parameters (at this stage).
- Doesn't seem to have any work (the instance variables can be initialised directly).
- But can be provided even if empty.

44 Copyright © 2004, Graham Roberts

Department of Computer Science



Classes and the main method

- We now have 2 classes, what about the main method and the code to use the classes?
- Options:
 - Write a test class and put the main method and code there.
 - Include the main method and code in the Library class.
 - Classes used in a larger program, main method somewhere else.

45 Copyright © 2004, Graham Roberts

Department of Computer Science



Library main

```
class Library
{
    // All the library stuff
    public static void main(String[] args)
    {
        // code here
    }
}
```

- Is this really a good idea?

46 Copyright © 2004, Graham Roberts

Department of Computer Science



Running the program

- Compile Library.java and Book.java
- Use the command:


```
java Library
```
- This will load Library.class and look for static main. If found it will be executed.

47 Copyright © 2004, Graham Roberts

Department of Computer Science



What goes in main?

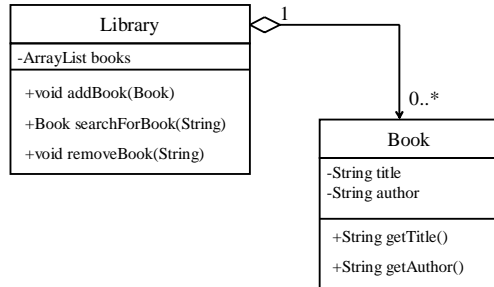
```
public static void main(String[] args)
{
    // Must create Library object here
    Library library = new Library();
    // Code to use the library follows
    ...
}
```

48 Copyright © 2004, Graham Roberts

Department of Computer Science



UML Class Diagram



49 Copyright © 2004, Graham Roberts

Department of Computer Science



Database

- Final point:
Couldn't all this have been done with a database?
- Yes!
- But that wouldn't have helped you learn Java!

50 Copyright © 2004, Graham Roberts

Department of Computer Science



However,

- We could have implemented the Library class using a database instead of Book objects.
- The public interface of Library (once finalised) would remain the same, the implementation could change.
- The power of encapsulation!

51 Copyright © 2004, Graham Roberts

Department of Computer Science



Summary

- A simple design example.
- Constructors and initialisation.
- Null reference.
- Classes and .java files.
- Using the main method.

52 Copyright © 2004, Graham Roberts

Department of Computer Science

