Introduction to Java (cs2514)

Lecture 1: Syllabus and Introduction

M. R. C. van Dongen

January 15, 2017

Syllabus

- Credit weighting 5.
- 24 x 1 hour lectures.
- Assessment: 100 marks.
    - Written exam 80 marks (1.5 hours).
    - Continuous assessment 20 marks: test and/or lab assignments.
    - Work that is submitted late shall be awarded zero marks.
- 40% pass rate.
- 1.5 hour repeat exam (mark for CA is carried forward).

Lectures

lecture	Monday 2 p.m. – 3 p.m. in WGB G03;
lecture	Friday 11 a.m. – 12 m. in WGB G01;
lab	Monday 4 p.m. – 6 p.m. in WGB G24;

Assignment Submission

- Submit your assignments using `http://cs4.ucc.ie/moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you’re too late for the official deadline,
  - You may still submit until the grace deadline.

Assignment Submission

- Submit your assignments using `http://cs4.ucc.ie/moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you’re too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline

Assignment Submission

- Submit your assignments using http://cs4.ucc.ie/moodle.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were "just" too late.
- When you're too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline,
  - You're too late.

Assignment Submission

- Submit your assignments using http://cs4.ucc.ie/moodle.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
    - Avoids problems for people who were “just” too late.
- When you’re too late for the official deadline,
    - You may still submit until the grace deadline.
- When you too late for the grace deadline,
    - You’re too late.
    - No submission allowed.

Assignment Submission

- Submit your assignments using `http://cs4.ucc.ie/moodle`.
- The official deadline is on the assignment sheet.
- There usually is a *grace* period, with a grace deadline.
  - Avoids problems for people who were “just” too late.
- When you’re too late for the official deadline,
  - You may still submit until the grace deadline.
- When you too late for the grace deadline,
  - You’re too late.
  - No submission allowed.
  - Sorry about that.

Module Content

- Class definitions;
- Procedural abstraction:
  - Every method should have a clear description that separates the method’s implementation from its users;
- Data abstraction:
  - Defining a data type with associated operations;
- Associations between objects;
- Class hierarchies and inheritance;
- Polymorphism and dynamic method binding.

Learning Outcomes

On Successful Completion Students should be able to

- Interpret a set of requirements for a software system;
- Construct Java programs in a good object oriented style;
- Design medium-sized software in a disciplined manner;
- Examine an existing software system for quality criteria;
- Employ object oriented abstractions such as encapsulation and inheritance in an appropriate way.

Recommended Literature

Loads of Good Java Books

title	Head First Java.
edition	Second;
author	Kathy Sierra & Bert Bates;
publisher	O'Reilly;
year	2004;
ISBN	978-0-596-00712-6.

- I strongly recommend this book.
- It's great if you already know how to program and want to learn a different language.

Recommended Literature (Continued)

Loads of Good Java Books

title	Effective Java;
author	Joshua Bloch;
publisher	Addison–Wesley;
year	2008;
ISBN	978-0-321-35668-0.

Recommended Literature (Continued)

Have Fun with Books

title	Java Puzzlers *Traps, Pitfalls, and Corner Cases;*
author	Joshua Bloch and Neal Gafter;
publisher	Addison–Wesley;
year	2005;
ISBN	0-321-33678-x.

Recommended Literature (Continued)

Advanced Stuff

title	Head First Object-Oriented Analysis & Design;
author	Brett D. McLaughlin, Gary Pollice, and David West;
publisher	O’Reilly;
year	2007;
ISBN	978-0-596-00867-3.

Plagiarism

Presenting Someone Else's Work as your Own

- When done deliberately, it is cheating.
- Plagiarism applies to:
  - Text;
  - Software;
  - Graphics;
  - Tables;
  - Formulae; or
  - Any representation of ideas in print, electronic or any other media.

UCC Policy on Plagiarism

- All students are required to read, to understand, and to comply with the UCC Policy on Plagiarism:
  - https://www.ucc.ie/en/media/support/recordsandexaminations/documents/UCC-Plagiarism-Policy---November-2017-V1.0---CLEAN.pdf.

Submitting Coursework

- In general, you should write all coursework in your own words.
- Coursework includes but is not limited to:
  - Programming assignments;
  - Literature reviews;
  - Abstracts and summaries;
  - Final-year projects.

Submitting Existing Software?

As a General Rule

assignments You are usually *not* allowed to submit existing software unless the lecturer clearly indicates that this is allowed.

- □ Consult with your course lecturer if you are unsure whether you are allowed to submit existing software for assignments.

project You are usually allowed to submit (small) parts of existing software.

- □ Consult with your project supervisor if you are unsure whether you are allowed to re-use existing software for your thesis.

Submitting Work from Others

- If you wish to quote small portions of software, text, include images, software, or other work created by others, you need to make it clear that you are doing so.
- You usually do this by putting quotation marks around quoted text and by including citations.
- Note that pictures and diagrams in books and papers may be copyrighted, in which case you need explicit permission from the copyright holder.

Acknowledging Existing Work

- If you acknowledge the original source, your lecturers/examiners will know that you are aware of the source.
    - You can receive credit for this in the form of marks.
- If you fail to acknowledge the source, your lecturers/examiners cannot give you any credit for using the source.
    - When failing to acknowledge the source is deliberate, this is a form of cheating, which may result in awarding a zero mark.

Citing Existing Software

- If you submit (parts of) existing software as part of your coursework, you should always give proper credit to the original author(s).
- In addition, you should clearly indicate which parts of these software are yours and which are not.
    - In a program listing you should indicate this using comments;
    - In a final-year project report you should also indicate the source of the software in the running text.
        - This should include a proper citation.

What is Java?

What is Java?

island: A holiday destination.

What is Java?

island: A holiday destination.

coffee: Nice one too.

What is Java?

island: A holiday destination.

coffee: Nice one too.

language: The programming language Java.

- □ We write our Java programs in text (source) files.

What is Java?

island: A holiday destination.

coffee: Nice one too.

language: The programming language Java.

- We write our Java programs in text (source) files.

compiler: We compile our source files with the javac compiler.

- Creates Java byte code file(s).
- Java byte code is an abstract machine language.
- Can be “run” with a (Java) *virtual machine* (VM).

What is Java?

island: A holiday destination.

coffee: Nice one too.

language: The programming language `Java`.

- □ We write our `Java` programs in text (source) files.

compiler: We compile our source files with the `javac` compiler.

- □ Creates Java byte code file(s).
- □ Java byte code is an abstract machine language.
- □ Can be “run” with a (Java) *virtual machine* (VM).

VM: We execute/interpret the byte code with the `java` VM.

- □ This “runs” the program.

How Does that Work?

Unix Session

```
$
```

How Does that Work?

Unix Session

```
$ ls
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$
```

How Does that Work?

Unix Session

```
$ ls
Hello.java
$ javac Hello.java
$ ls
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
Hello.java    Hello.class
$
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
Hello.java    Hello.class
$  java Hello
```

How Does that Work?

Unix Session

```
$  ls
Hello.java
$  javac Hello.java
$  ls
Hello.java    Hello.class
$  java Hello
Hello world!
$
```

State and Behaviour

- Classes *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

Knowing: What the object knows.

Doing: What the object does.

State and Behaviour

- Classes *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

Knowing: What the object knows.

- The object's *instance variables* define its memory.

Doing: What the object does.

State and Behaviour

- Classes *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

Knowing: What the object knows.

- The object's *instance variables* define its memory.
- Instance variables are also known as *attributes*.

Doing: What the object does.

Syllabus
Literature
Plagiarism
Using Java
Classes and Objects
State and Behaviour
Class versus Object
Object Analogy
Example
The Class
Exercise
Question Time
For Next Friday
Acknowledgements
References
About this Document

State and Behaviour

- Classes *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.

  - The object's *instance variables* define its memory.
  - Instance variables are also known as *attributes*.
  - The object's attributes define the object's *state*.

  Doing: What the object does.

State and Behaviour

- Classes *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

  Knowing: What the object knows.

  - The object's *instance variables* define its memory.
  - Instance variables are also known as *attributes*.
  - The object's attributes define the object's *state*.

  Doing: What the object does.

  - What object do "are" its (instance) *methods*.

State and Behaviour

- Classes *define* your objects.
- Each class corresponds to an *object type*.
- When you create a class, you think about:

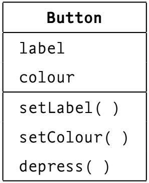
Knowing: What the object knows.

- The object's *instance variables* define its memory.
- Instance variables are also known as *attributes*.
- The object's attributes define the object's *state*.

Doing: What the object does.

- What object do "are" its (instance) *methods*.
- The object's instance methods define its *behaviour*.

State and Behaviour

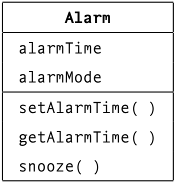


Button

label
colour

setLabel( )
setColour( )
depress( )

State and Behaviour



Alarm

alarmTime
alarmMode

setAlarmTime( )
getAlarmTime( )
snooze( )

How Does it Work in Java?

Java

```
public class Song {
    private final String title;  // Attribute: knowing.
    private final String artist; // Attribute: knowing.

    public String getTitle( ) { // Method: doing.
        return title;
    }

    public void playSong( ) {   // Method: doing.
        // play song.
    }
    ⋮
}
```

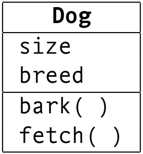
Class versus Object

- A class is not an object.
- A class constructs objects at run time.
- The class acts as a *blueprint* for the object.
- One class may define several objects (a.k.a. instances).

Class versus Object

One Class

- A class is not an object.
- A class constructs objects at run time.
- The class acts as a *blueprint* for the object.
- One class may define several objects (a.k.a. instances).



Dog

size
breed

bark( )
fetch( )

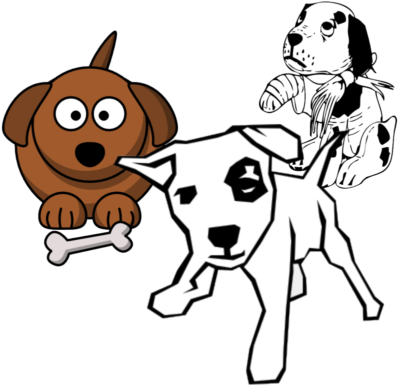
Class versus Object

Many Instances

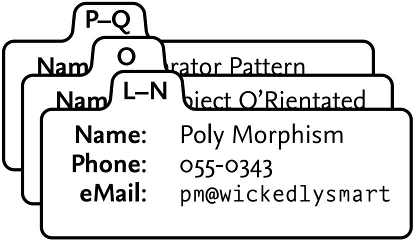
- A class is not an object.
- A class constructs objects at run time.
- The class acts as a *blueprint* for the object.
- One class may define several objects (a.k.a. instances).

Dog
size breed

bark( )
fetch( )

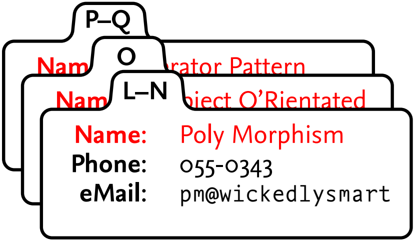


Objects are like Rolodex Cards



Objects are like Rolodex Cards

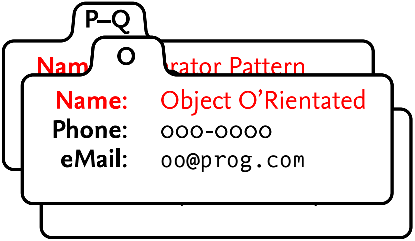
Name



Name:	Poly Morphism
Phone:	055-0343
eMail:	pm@wickedlysmart

Objects are like Rolodex Cards

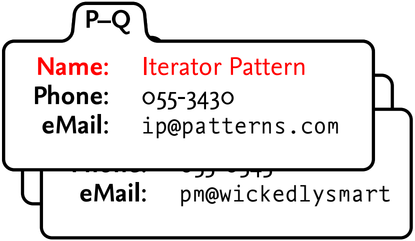
Name



Name: Object O’Rientated
Phone: 000-0000
eMail: oo@prog.com

Objects are like Rolodex Cards

Name



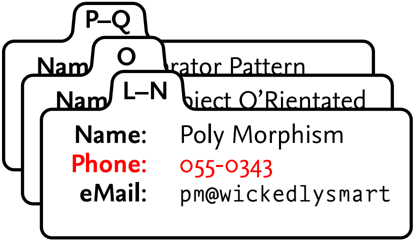
P–Q

Name: Iterator Pattern
Phone: 055-3430
eMail: ip@patterns.com

eMail: pm@wickedlysmart

Objects are like Rolodex Cards

Phone



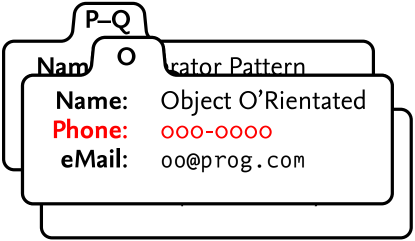
Name: Poly Morphism

Phone: 055-0343

eMail: pm@wickedlysmart

Objects are like Rolodex Cards

Phone



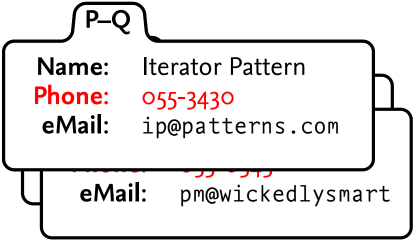
Name: Object O’Rientated

Phone: 000-0000

eMail: `oo@prog.com`

Objects are like Rolodex Cards

Phone



P–Q

Name: Iterator Pattern

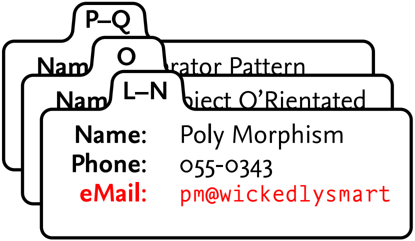
Phone: 055-3430

eMail: ip@patterns.com

eMail: pm@wickedlysmart

Objects are like Rolodex Cards

eMail



P–Q

Nam rator Pattern

O

Nam ject O’Rientated

L–N

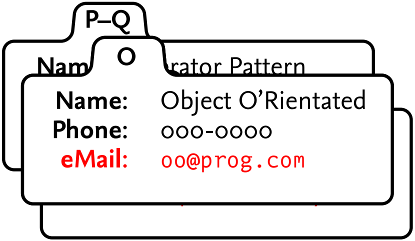
Name: Poly Morphism

Phone: 055-0343

eMail: `pm@wickedlysmart`

Objects are like Rolodex Cards

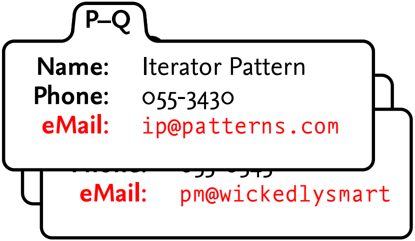
eMail



Name: Object O'Rientated
Phone: 000-0000
eMail: `oo@prog.com`

Objects are like Rolodex Cards

eMail



P–Q

Name: Iterator Pattern
Phone: 055-3430
eMail: ip@patterns.com

eMail: pm@wickedlysmart

How Does it Work in Java?

The main() Defines the Main Execution Thread

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

The Dog Instance Attibutes

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

The Types of the Attributes

Java

```java
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```java
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Create new Dog

Java

```java
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```java
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set John's Breed

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set John's Size

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Create new Dog

Java

```java
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```java
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set Lucky's Breed

Java

```
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Set Lucky's Size

Java

```java
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```java
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Make John Bark

Java

```java
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```java
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

How Does it Work in Java?

Make Lucky Bark

Java

```java
public class Dog {
  public String breed;
  public double size;


  public void bark( ) {
    if (size > 1.0) {
      System.out.println( "Ruff" );
    } else {
      System.out.println( "Bark" );
    }
  }
}
```

Java

```java
public class Main {
  public static void main( String[] args ) {
    Dog john = new Dog( );
    john.breed = "Bulldog";
    john.size = 2.0;
    Dog lucky = new Dog( );
    lucky.breed = "Terrier";
    lucky.size = 0.5;

    john.bark( ); // Ruff
    lucky.bark( ); // Bark
  }
}
```

The Class

- Every class has a unique object that represents the class.
- These objects have attributes.
  - We call them *class attributes*.
- Class and instance attributes are not the same.
  - Each instance of the class owns its instance attributes.
  - Each class owns its class attributes.
  - The class to class attribute relationship is one-to-one.
  - The class to instance attribute relationship is one-to-many.
  - Means you cannot use class attributes to represent object state.
- An object can access its class' class attributes.
- You declare a class attribute by adding the word `static`.
- There are also class methods (add `static`).

Example

Java

```java
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

More About that in another Lecture

Java

```
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Class Attribute

Java

```
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Instance Attributes

Java

```
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Constructor

Java

```
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Instance Method

Java

```java
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Class Method

Java

```java
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Class Method

Java

```java
public class Beer {
    private static final Double DISCOUNT = 0.50; // class attribute
    private final String name; // instance attribute
    private final double price; // instance attribute

    // Constructor
    public Beer( final double price, final String name ) {
        this.name = name;
        this.price = price;
    }

    // Instance Method
    public void sell( ) {
        final double sellingPrice = price - DISCOUNT;
        System.out.println( "Selling " + name + " at " + sellingPrice );
    }

    // Class Method
    public static void main( String[] args ) {
        final Beer miDaza = new Beer( 5.00, "Mi Daza" );
        final Beer redemption = new Beer( 5.60, "Redemption" );
        System.out.println( "Selling beers with " + DISCOUNT + " discount." );
        miDaza.sell( );
        redemption.sell( );
    }
}
```

Executing the main

Unix Session

```
$
```

Executing the main

Unix Session

```
$ ls
```

Executing the main

Unix Session

```
$  ls
Beer.java
$
```

Executing the main

Unix Session

```
$  ls
Beer.java
$  javac Beer.java
```

Executing the main

Unix Session

```
$  ls
Beer.java
$  javac Beer.java
$
```

Executing the main

Unix Session

```
$  ls
Beer.java
$  javac Beer.java
$  ls
```

Executing the main

Unix Session

```
$  ls
Beer.java
$  javac Beer.java
$  ls
Beer.java    Beer.class
$
```

Executing the main

Unix Session

```
$  ls
Beer.java
$  javac Beer.java
$  ls
Beer.java    Beer.class
$  java Beer
```

Executing the main

Unix Session

```
$  ls
Beer.java
$  javac Beer.java
$  ls
Beer.java    Beer.class
$  java Beer
Selling beers with 0.5 discount.
Selling Midaze at 4.5
Selling Redemption at 5.1
$
```

Exercise

- Let’s implement an `Account` class for bank accounts.
- For simplicity every `Account` object has:
  - A `balance` attribute for the current balance;
  - A `number` attribute for the account number;
  - A `transactions` attribute for counting the transactions.
- When an account is created, the bank gives an initial credit of 1 Euro.
- There is a `debit( )` instance method for debiting the account.
- For simplicity there won’t be other methods.
  - You may implement them when there’s more time.
- The class should be easy to use and easy to maintain.
  - If you manage to implement a class like that, well done…
- You have 5 minutes.

Question Time

Questions Anybody?

For Next Friday

- Study the Java part of the presentation.
- Implement one of the Java programs.
- Compile the program.
- Fix errors if you have them.
- Run the program.

Acknowledgements

- This lecture is partially based on
  - [Sierra, and Bates 2004, Chapter 1].

Bibliography I

Sierra, Kathy, and Bert Bates [2004]. *Head First Java*. O'Reilly. ISBN: 978-0-596-00712-6.

About this Document

- This document was created with `pdflatex`.
- The LaTeX document class is `beamer`.