Middlesex University London

Lecture 10

Classes and Objects

THIS WEEK

- OOP
- Defining Classes and Creating Objects
- Constructing Objects Using Constructors
- Accessing Objects
- Visibility Modifiers
- Data Field Encapsulation
- Passing Objects to Methods
- The this Reference

INTRODUCTION

- What is a software system?
- How do we design/maintain such a system?
- Why are there difficulties designing and maintaining?
- How can we overcome these difficulties?

COMPOSITION

- Break down into manageable pieces
  - Smaller components – easier to deal with

DESIGN APPROACH: IDENTIFY ENTITIES WITHIN A SYSTEM

- Building a system around the object with in
- "real world" entities that have meaning with in the context of the system description
- Once potential objects are identified, try to recognise their characteristics in terms of
  - attributes (data)
  - possible operations on those data items (think of "verbs")
- In a sense, this type of approach concentrates on recognising structural aspects of a system first, followed by behavioral aspects

DESIGN APPROACH: IDENTIFY RESPONSIBILITIES

- Begins with an analysis of behaviour in a system
  the argument goes that behaviour is understood very early on
- –"… an object-oriented program is a community of interacting agents called objects, each with a role to play."
- –objects are senders and receivers of messages which either request or return the result of a service respectively

- •Think of one „server" object as providing a service or set of services which other „client" objects can request

OBJECT-ORIENTED SYSTEMS

- OOP is essentially a technology for developing **reusable software.**
- Composite, modular and organised around data
- They map to the way we see the world
- Best strategy for large, evolving software systems (**Scalability**)
- Also enforce designing before coding
- **Maintainability**
- **Extensibility**

OBJECTS

- What is an "Object"
  - …Self contained item that represents something we want to model
  - An Instance of a Class
  - Examples could be……..?

THREE OBJECT CHARACTERISTICS

- State
- Behaviour
- Identity

STATE

- **Think of data as items of information contained in an object**
- Also known as its properties or attributes. It is represent by data fields with their current value.

- A Circle object may have data fields such as *radius or colour*.
- A Rectangle object has data field *width* and *heights*.
- An Account object has data fields *id, balance, annual intrest rate*

BEHAVIOUR

- **The behaviour of an object is defined by the operations that can "act" on its data**
- The behaviour of an object (also know an its actions) defined by methods.
- To invoke method on an object is to ask the object to perform action.
- An object can have a set of actions that it can perform.
  - The behaviour may be commanded from outside the object
    - **Command** – the object is commanded to do something (Example: *set Radius,* set Id)
    - **Query** – the object is requested for something (Example: *getRadius or get Id)*

IDENTITY

- **An object can be identified**
  - you know what it is (will usually be some other object knows what it is in a running system)
  - it knows what it is
  - Examples might be
    - Circle could be bigCircle
    - Rectangle could be a square
    - Account could be basicAccount

CLASSES IS A TEMPLATE OF OBJECT (BLUEPRINT)

- A class is a template, blueprint or contract tat defines what objects data fields and methods will be

Circle
radius: double
getArea() : double getPerimeter(): double setRadius(newR : double): void

```
 */
public class Circle {

    double radius;

    double getArea (){
        return radius * radius * Math.PI;
    }

    void setRadious(double newR){
        radius = newR;
    }

}
```

Data field

Methods

```
 */
public class Circle {

    double radius;

    public double getArea (){
        return radius * radius * Math.PI;
    }

    public void setRadious(double newR){
        radius = newR;
    }

}
```

separate "driver" class

```
public class CircleTester {

    public static void main(String[] args) {

        Circle c =  new Circle();
        c.setRadious(10);
        System.out.printf("%.2f",c.getArea());
    }

}
```

THE DOT OPERATOR

- The dot operator (.) gives you access to an object's state and behaviour (instance variables and methods)
- •In order to use the dot operator you must first have created an object and given it an "identity" . . .

```
public class CircleTester {

    public static void main(String[] args) {

        Circle c =  new Circle();
        c.setRadious(10);
        System.out.printf("%.2f",c.getArea());
    }

}
```

CONSTRUCTOR

- A constructor is invoked to create an object using the *new* key word
- Every class have a default constructor
- A constructor must have the same name as the class itself
- Constructor plays the role of initialising an object

```
public class CircleTester {

    public static void main(String[] args) {

        Circle c = new Circle();
        c.setRadious(10);
        System.out.printf("%.2f",c.getArea());
    }

}
```

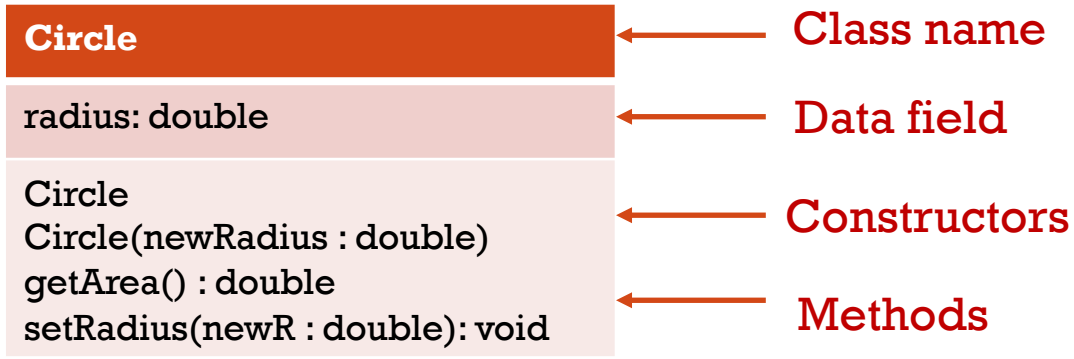
```
public class Circle {

    double radius;

    Circle(){

    }
    Circle(double newRadius){
        radius = newRadius;
    }

    double getArea (){
        return radius * radius * Math.PI;
    }

    void setRadious(double newR){
        radius = newR;
    }

}
```

Constructors

```java
public class Circle {

    double radius;

    Circle(){

    }
    Circle(double newRadius){
       radius = newRadius;
    }

    double getArea (){
        return radius * radius * Math.PI;
    }

    void setRadious(double newR){
        radius = newR;
    }

}
```

```java
public class CircleTester {

    public static void main(String[] args) {

        Circle c =  new Circle();
        c.setRadious(10);
        System.out.printf("%.2f",c.getArea());

        Circle c2 = new Circle(100);
        System.out.printf("\n%.2f",c2.getArea());
    }

}
```

UML

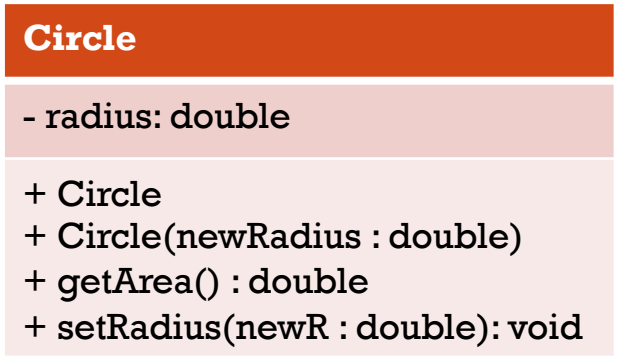

Circle
radius: double
Circle
Circle(newRadius : double)
getArea() : double
setRadius(newR : double): void
Class name
Data field
Constructors
Methods

VISIBILITY MODIFIERS

Access Levels

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

UML



Class example:

```java
public class Circle {

    private double radius;

    public Circle(){

    }
    public Circle(double newRadius){
        radius = newRadius;
    }

    public double getArea (){
        return radius * radius * Math.PI;
    }

    public void setRadious(double newR){
        radius = newR;
    }

}
```

GETTER AND SETTER

In **Java accessors** are used to
get the value of a **private field**
and **mutators** are used to set the value of a **private field**.

Accessors are also known as **getters**
and **mutators** are also known as **setters**

```java
public class Circle {

    private double radius;

    public Circle(){

    }
    public Circle(double newRadius){
       radius = newRadius;
    }
    public double getArea (){
        return radius * radius * Math.PI;
    }

    public void setRadious(double newR){
        radius = newR;
    }
     public double getRadious(){
        return radius;
    }

}
```

THIS

this keyword in Java can be used inside the m*ethod* or *constructor* of a class.
It(***this)*** works as a reference to the current object, whose method or constructor is being invoked.

This keyword can be used to refer to any member of the current object from within an instance method or a constructor.

```java
public class Circle {

    private double radius;

    public Circle(){

    }
    public Circle(double radius){
       this.radius = radius;
    }
    public double getArea (){
        return radius * radius * Math.PI;
    }

    public void setRadious(double radius){
        this.radius = radius;
    }
     public double getRadious(){
        return radius;
    }

}
```

KEY THINGS YOU SHOULD UNDERSTAND AFTER WEEK 10

- OOP
- Defining Classes and Creating Objects
- Constructing Objects Using Constructors
- Accessing Objects
- Visibility Modifiers
- The this Reference

MASTERING YOUR SKILLS

- Read chapter 9 from the book and go over the exercises in the end of the chapter.