CSU11012 - Introduction to Programming II

Hitesh Tewari

131 Lloyd Institute

http://mymodule.tcd.ie

- Two Lectures and one Tutorial each week
  - Lecture
  - Lecture
  - Tutorial

- One Lab session each week

Assessment

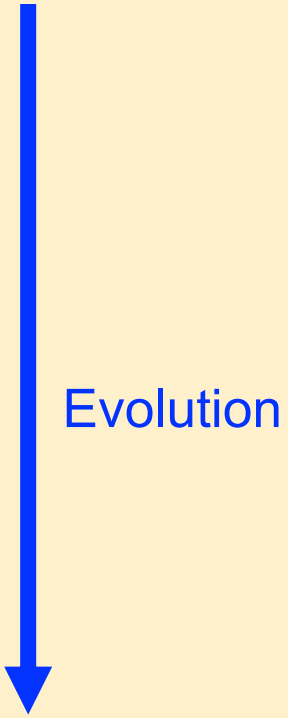
- Laboratories (20%)
  - Four Labs worth 5% each
    - Due on weeks 4, 6, 9 & 11
- eTest (80%)
  - 2hr eTest with two programming questions

Aims

- Provide an introduction to the object-oriented approach to program design (OOP)
  - Teaches you how to write programs in an object-oriented language (in this case Java)
- Opportunity to reinforce your problem solving and programming skills
  - By developing solutions to programming problems

Programming Techniques

- Unstructured Programming
- Structured Programming
- Modular Programming
- Object-Oriented Programming



Unstructured Programming

- Writing small programs consisting of only one **main** program which consists of a sequence of statements
  - Modify data which is global throughout the program
- If the same statement sequence is needed at different locations within the program

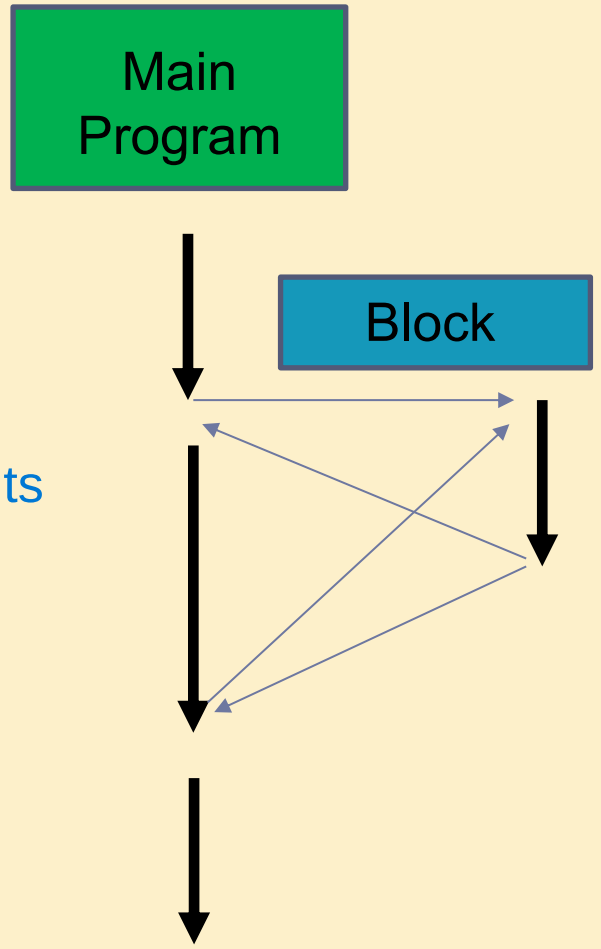
  Sequence must be copied
- If an error needs to be modified

  Every copy must be modified

Structured Programming

- Combine sequences of calling statements into one place
- Make use of flow-control, loops etc.

  No arbitrary jumps such as GOTO statements
- Programs can be written in a more structured manner and error free

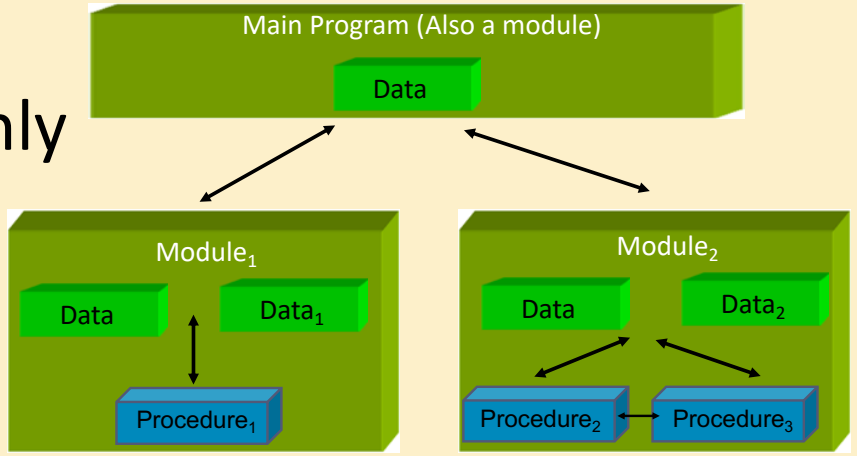


Modular Programming

- Subdividing your program into separate subprograms
  - Procedures of a common functionality are grouped together into modules

- Program no longer consists of only one part

  Divided into smaller parts which interact through procedure shells



- Main program coordinates calls to procedures in separate modules

  Hands over appropriate data as parameters

What is OOP?

- Object oriented programming (OOP) involves programming using objects

- An *object* represents an entity that can be distinctly identified
  - e.g. a laptop, a desk, a chair
  - Allows us to do things more efficient
  - Can be re-used
- The attributes (colour, size etc.) are known as
  - The state of an object

- Its functionality, such as “write” for a pen
  - Is known as behaior
  - Methods

Example Object

- Object

  - House

- State

  - Location, colour, area of house

- Behaviour

  - Close/open main door

Reference Types (Classes)

- Let's imagine you are creating a program that stores info about someone doing CS
- You might create a set of variables (instances of types) as follows:

```
String forename = "Kay";
String surname  = "Oss";
```

- Now you need to add two people so you add:

```
String forename2 = "Don";
String surname2  = "Keigh";
```

- Now you need to add more – this is getting messy

Better if we create our own type & call it "Student"

Problem:

Create a class Student and give it some meaningful attributes.

Reference Types (Classes)

- In Java we create a **class** which acts as a blueprint for a custom type

```
public class Student
{
    String forename;
    String surname;
}
```

- A class
  - Has *attributes*

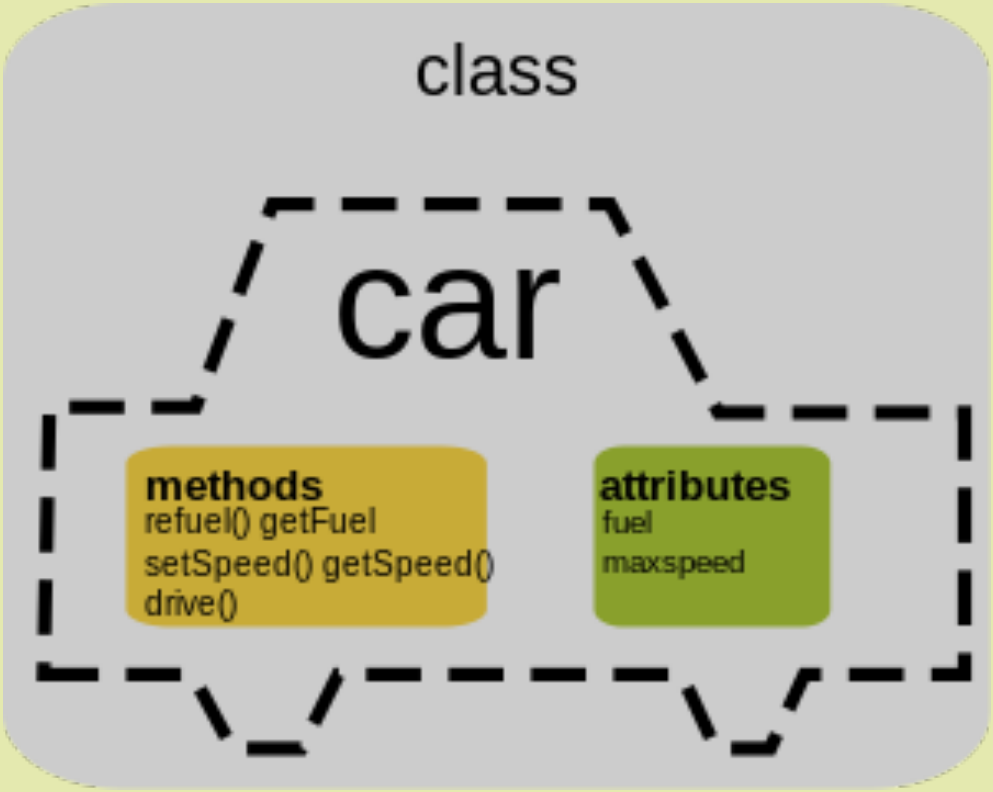
    Things it is assigned e.g. name, age

  - It also has *methods*

    Things it can do e.g. walk, think

Problem:

Create a reference class Car from the image shown below.



Blackboard: Week1/Car1

```
public class Car
{
    private String fuel;
    private int maxSpeed;

    public void refuel(int liters)
{
}

    public String getFuel(String typeOfFuel)
{
}

    public void setSpeed(int speed)
{
}

    public int getSpeed()
{
}

    public void drive()
{
}
}
```

Problem:

Create a `Student` class and write methods to add and retrieve the `name`, `age` and `course` of a student.

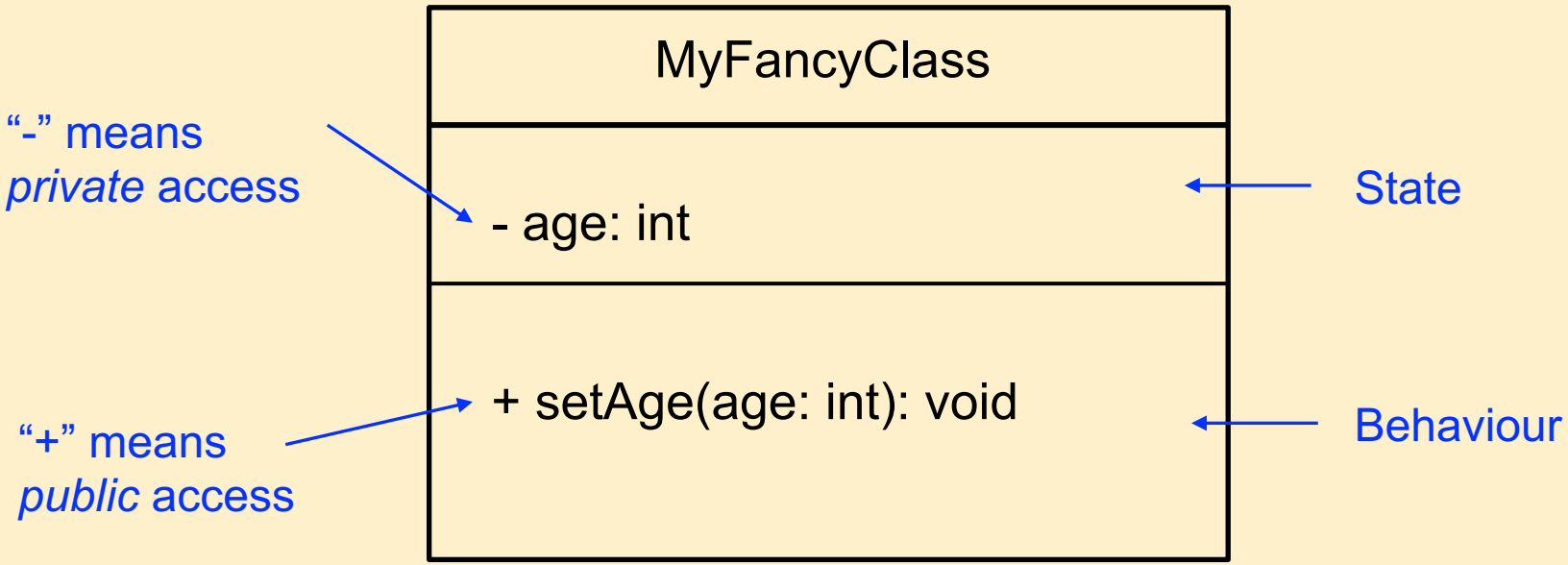
Blackboard: Week1/Student2

```
public class Student
{
    private String name;
    private int dob;
    private String course;

    public String getName()
    {
        return name;
    }

    public void setName(String studentName)
    {
        name = studentName;
    }
}
```

A Class in Java (cont.)

- A class in Java can contain
  - Attributes
  - Methods
  - Constructors
  - Blocks
  - Other Classes & Interfaces

Representing a Class Graphically (UML)



Objects

- When we create an **instance** of a class we
  - Assign memory to hold the attributes
  - Assign the attributes
- We call the instance an **object**

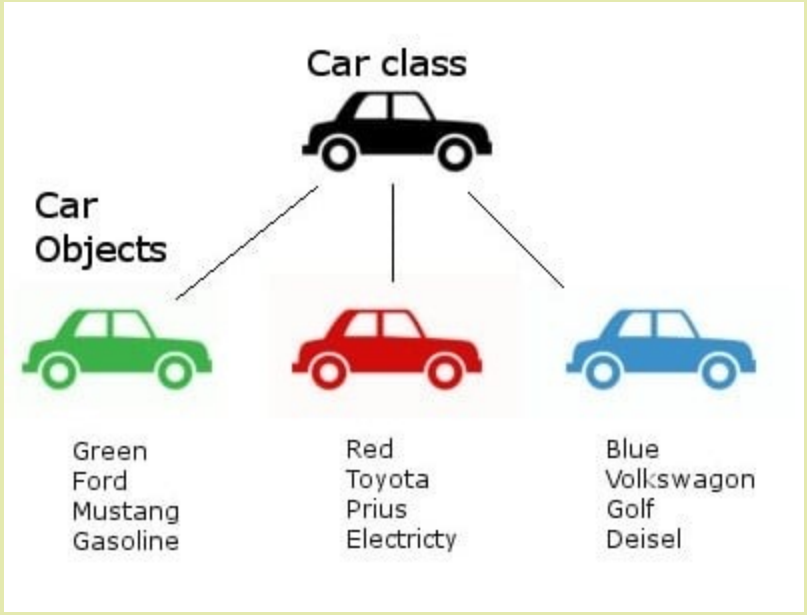
Anatomy of an OO Program

Blackboard: Week1/MyFancyClass

```
public class MyFancyClass
{
    //Class state (properties that an object has such as colour or size)
    private int age;

    public void setAge(int a)
    {
        //Class behaviour (actions an object can do)
    }

    public static void main(String[] args)
    {
        //Create an object of type MyFancyClass in memory and get a reference to it
        MyFancyClass c = new MyFancyClass();
    }
}
```

Problem:

Define a class Car and create three objects of class Car as depicted in the image below.



Blackboard: Week1/Car2

```
public class Car
{
    private String color;
    private String manufacturer;
    private String model;
    private String fuel;
}


public static void main(String[] args)
{
    Car ford = new Car();
    Car toyota = new Car();
    Car volkswagon = new Car();
    for
}
```