CT2106
Object Oriented Programming

Dr. Frank Glavin
Room 404, IT Building
Frank.Glavin@University*of*Galway.ie
School of Computer Science

University *of* Galway.ie

Summary of Last Two Lectures

- A class has fields, a constructors and methods
- Encapsulation - each object's data (fields) is protected by its accessor/mutator methods
- If you want to access/change an object's state, you must use its accessor/mutator methods
- The use of the 'private' key word prevents external access to an object's data
- Java is both compiled and Interpreted
- Java uses JVM to execute the same code on multiple platforms/machines

Ollscoil na Gaillimhe
University of Galway

Today's Lecture

- How to implement a scenario?
- An object can be composed of other objects
- Objects can call each other's methods
- Java uses Reference types as well as primitive types
- What to watch out for in Integer division
- To use double and boolean primitive values
- To use conditional statements

Ollscoil na Gaillimhe
University of Galway

An Example Problem to Solve/Implement

We wish to be able to create several Car (objects)

Each car object has an Engine

Each Engine has the following properties

- **kpg** (kilometers per gallon)
- **fuel** (amount of fuel in the tank)

Each Car has a **totalDistance** (travelled)

Ollscoil na Gaillimhe
University of Galway

Problem

Each Car should have a **move method** specifying the distance to be travelled

You may call this method as often as you wish, and the car will print out

- Total distance travelled so far
- Remaining fuel
- Estimated distance left to travel

If you are out of fuel, the car will notify you

Ollscoil na Gaillimhe
University of Galway

How to Start

Firstly, identify the classes

Code up the basic classes

Remember each class should have

- Fields
- At least one constructor
- Methods

Ollscoil na Gaillimhe
University of Galway

Linking classes

Each Car object “**has a**”/ “**has an**” Engine

In OOP terms, this means that a Car object relies upon the service of an Engine object

Ollscoil na Gaillimhe
University of Galway

Is-a vs **has-a** relationships

- Two fundamental relationships between classes in OOP
  - **has-a** (or composition)
  - **is-a** (or inheritance) : we'll encouter this later

- A RacingBike is-a type of Bicycle (Inheritance)
- A RacingBike has-a Wheel (Composition)

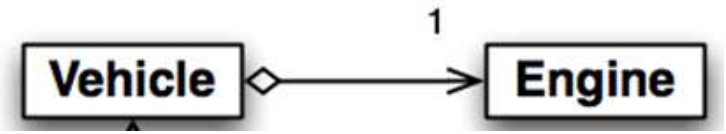
Ollscoil na Gaillimhe
University of Galway

Representing **has-a** relationships

- **has-a** relationship denotes **composition**
- One object is **composed** of another and relies upons its services for its own functionality
- A Vehicle **has-a**(n) Engine; A Bicycle has a wheel

Representing **has-a** relationships

- In OOP class diagrams a diamond shape like this indicates a composition or has-a relationship



- This class diagram tell us that a Vehicle object is composed of a single Engine object

Ollscoil na Gaillimhe
University of Galway

Realising composition in Java

- To realise a has-a relationship you have to create a link between the participant classes
- You do this using a new type of variable type: **a reference variable type**
- The reference declaration is in the **owner** class
- In our example, the Car class will have reference variable that points to an Engine object

Ollscoil na Gaillimhe
University of Galway

```
public class Bicycle
{
    // instance variables - replace the example below
    private int speed;
    private int gear;
    private int cadence;
    private Wheel front;
    private Wheel back;

    /**
     * Constructor for objects of class Bicycle
     */
    public Bicycle(int speed, int gear, int cadence)
    {
        // initialise instance variables
        this.speed = speed;
        this.gear = gear;
        this.cadence = cadence;
        front = new Wheel(5);
        back = new Wheel (5);
    }
```

Two reference variable of type Wheel are declared

The variables are initialised in the constructor

Wheel Class

```
public class Wheel
{
    // instance variables - replace the example below with
    private int radius;

    /**
     * Constructor for objects of class Wheel
     */
    public Wheel(int radius)
    {
        // initialise instance variables
        this.radius = radius;
    }   }
```

Following this example, you can create a link between Car and Engine

Ollscoil na Gaillimhe
University of Galway

Information Required

- What information does the Car object require from Engine object?
  - "Each car should have a **move method** specifying the distance to be travelled"

- You may call this method as often as you wish and the Car will print out:
  - Total distance travelled so far
  - Remaining fuel
  - Estimated distance left to travel

  "If you are out of fuel, the car will notify you"

Ollscoil na Gaillimhe
University of Galway

Objects Communicating

- What information does the Car object require from Engine object?
-
- We know this
  - Engine object has:
    - ❖ Fuel amount
    - ❖ kpg (the amount of fuel used per distance)

- Car object has
  - The distance amount
  - The total distance travelled amount
  - A move method

Ollscoil na Gaillimhe
University of Galway

Car to Engine

- Car has no information about fuel levels
- It requires Engine to give it that

Engine to Car

- Engine has no information about distance
- It requires Car to give it this (so that it can calculate fuel consumption)

Ollscoil na Gaillimhe
University of Galway

go(int distance) method in Engine class

```java
/**
 * go method of the engine calculates the amount of fuel needed to go
 * the distance required. It updates the fuel variable based on this calculation.
 * It returns false if the updated fuel calculation is less than zero.
 * This is a rough and ready way to determine if the fuel level can accomodate the distance required.
 * Can you do better ? For example, if there was fuel for 5 km, but the distance variable was 10km
 * perhaps this method should return the distance that could be travelled, rather
 * than returning false.
 *
 * @param  distance  : the distance required to travel
 * @return  true or false based on whether it is possible or not
 */
public boolean go(int distance)
{
    fuel = fuel - distance/kpg; // integer division problem here. Can you spot it?
    if(fuel >=0){
        return true;
    }

    return false;
}
```

Ollscoil na Gaillimhe
University of Galway

setFuel(int fuel) from the **Car** class

```
public void setFuel(int fuel){
    engine.setFuel(fuel);
}
```

Ollscoil na Gaillimhe
University of Galway

move(int distance) from the Car class

```
/**
 * The move method is called whenever a Car object is required to move
 *
 * @param  distance  : the distance the car wishes to move
 * @return   boolean: true or false based on whethe the car moved or not
 */
public boolean move(int distance)
{

    boolean moved = engine.go(distance); //checks to see if engine will allow this distance

    if(moved){
        totalDistance +=distance; //updates distance travelled
    }

    return moved;
}
```

Ollscoil na Gaillimhe
University of Galway

First Assignment

- Based on this example sand will be posted later today.
- It will be due next Friday.

Ollscoil na Gaillimhe
University of Galway