جامعة الجلالة
GALALA UNIVERSITY

Course: Object oriented

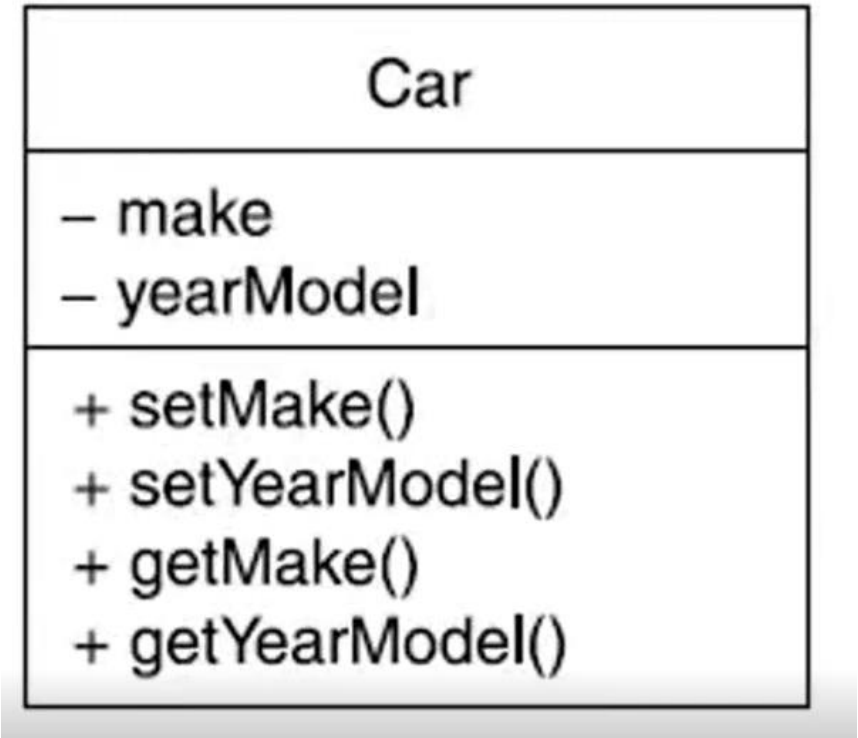
Grade: Firth year

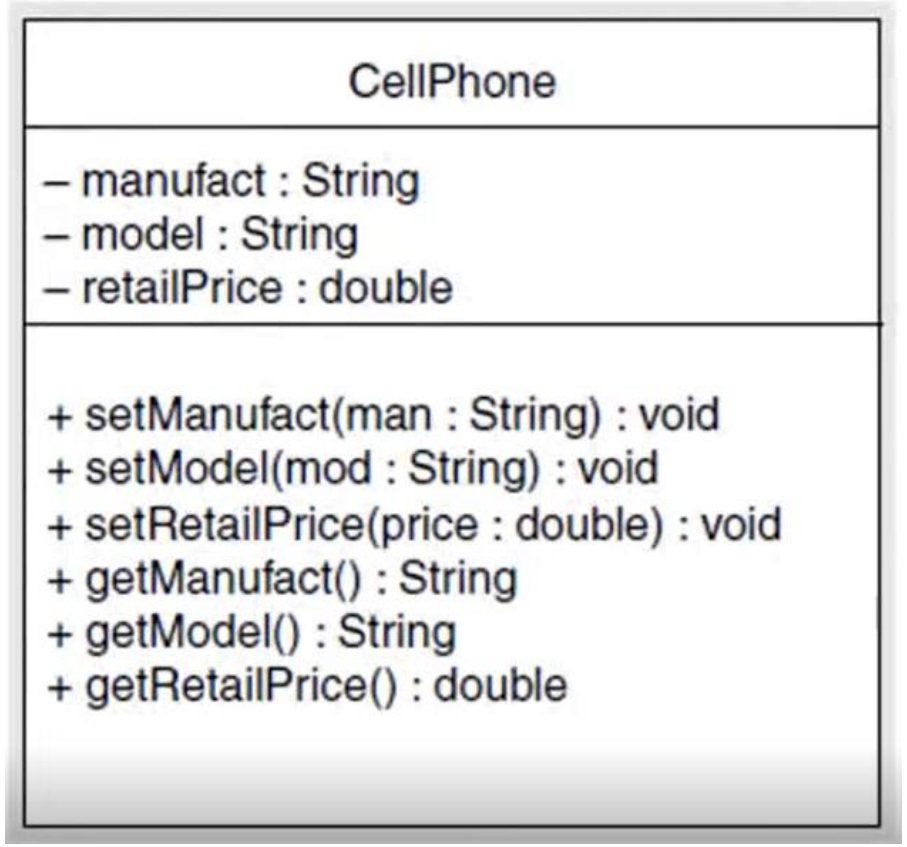
Assistant Lecturer : Yasmin Alsakar at Faculty of Computer & Information Sciences - Mansoura University.

OUTLINE

- **Constructors**
- **Constructors – Initialization list**
- **Overloading Methods and Constructors**
- **Constructor Overloading**

Other Examples





Car example

```java
package projectoop1;

public class Car {
    private String maker;
    private int model;

    public void setmaker(String m)
    {
        if (m == "Toyta" || m == "Honda" || m == "Merceds")
            maker = m;
        else
            System.out.println("Invalid Maker");
    }
```

```java
public void setmodel(int year)
{
    if(year > 2010)
        model = year;
    else
        System.out.println("Invalid Model");
}
```

```java
    public String getmaker()
    {
        return maker;
    }
    public int getmodel()
    {
        return model;
    }
}
```

Main function

```java
package projectoop1;

public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Car c1;
        c1 = new Car();
        Car c2 = new Car();
        c1.setmaker("Honda");
        c1.setmodel(2017);
        c2.setmaker("Toyta");
        c2.setmodel(2018);
        System.out.println(c1.getmaker() + " " + c1.getmodel());
        System.out.println(c2.getmaker() + " " + c2.getmodel());
    }
}
```

Constructors

- Classes can have special methods called *constructors*.
- A constructor is a method that is automatically called when an object is created. Recatngle r1; -------- Car c1;
- Constructors typically initialize object attributes and perform other object initialization tasks.
- Constructors are used to perform operations at the time an object is created.

- Constructors have a few special properties that set them apart from normal methods.
  - Constructors have the same name as the class.
  - Constructors have no return type (not even `void`).
  - Constructors may not return any values.
  - Constructors are typically public.

```
public :
      Rectangle( )
    {
       length = 0;
       width =  0;
    }
```

Example for constructor:

- For Rectangle:

```
public Rectangle ()
{
    length = 10;
    width = 15;
}
```

```java
package projectoop1;

public class Rectangle {
    private double length;
    private double width;

    public Rectangle ()
    {
        length = 10;
        width = 15;
        System.out.println("a new room created with 10 m length and 15 width");
    }
    public Rectangle(double l, double w)
    {
        length = l;
        width = w;
    }
```

```java
    public void setLength(double l)
    {
        length = l;
    }
    public void setWidth(double w)
    {
        width = w;
    }

    public double getLength()
    {
        return length;
    }

    public double getWidth()
    {
        return width;
    }

    public double getArea()
    {
        return length*width;
    }
}
```

Main function

```java
public class ProjectOOP1 {

    public static void main(String[] args) {
        // TODO code application logic here
        Rectangle r1 = new Rectangle();
        System.out.println(r1.getLength());
        System.out.println(r1.getWidth());

        r1.setLength(25);
        r1.setWidth(50);
        System.out.println(r1.getLength());
        System.out.println(r1.getWidth());

        Rectangle r2 = new Rectangle(40, 60);
        System.out.println(r2.getLength());
        System.out.println(r2.getWidth();

    }
}
```

Overloading Methods and Constructors

- Two or more methods in a class may have the same name as long as their signatures are different.
- Method signature (No of Args – Types of Args – Order of Args)
- When this occurs, it is called *method overloading*. This also applies to constructors.
- Method overloading is important because sometimes you need several different ways to perform the same operation.

```
int add(int num1, int num2)
{
   int sum = num1 + num2;
   return sum;
}

int add(int num1, int num2, int num3)
{
   int sum = num1 + num2 + num3 ;
   return sum;
}

Float add(float num1, float num2)
{
        float sum = num1 + num2;
        return sum;
}
```

Constructor Overloading

```
Rectangle::Rectangle ( ):length(0),width(0)
{
}

Rectangle::Rectangle(float l , float w):length(l),width(w)
{
}
```

Thank you