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**

Overloading Methods and Constructors

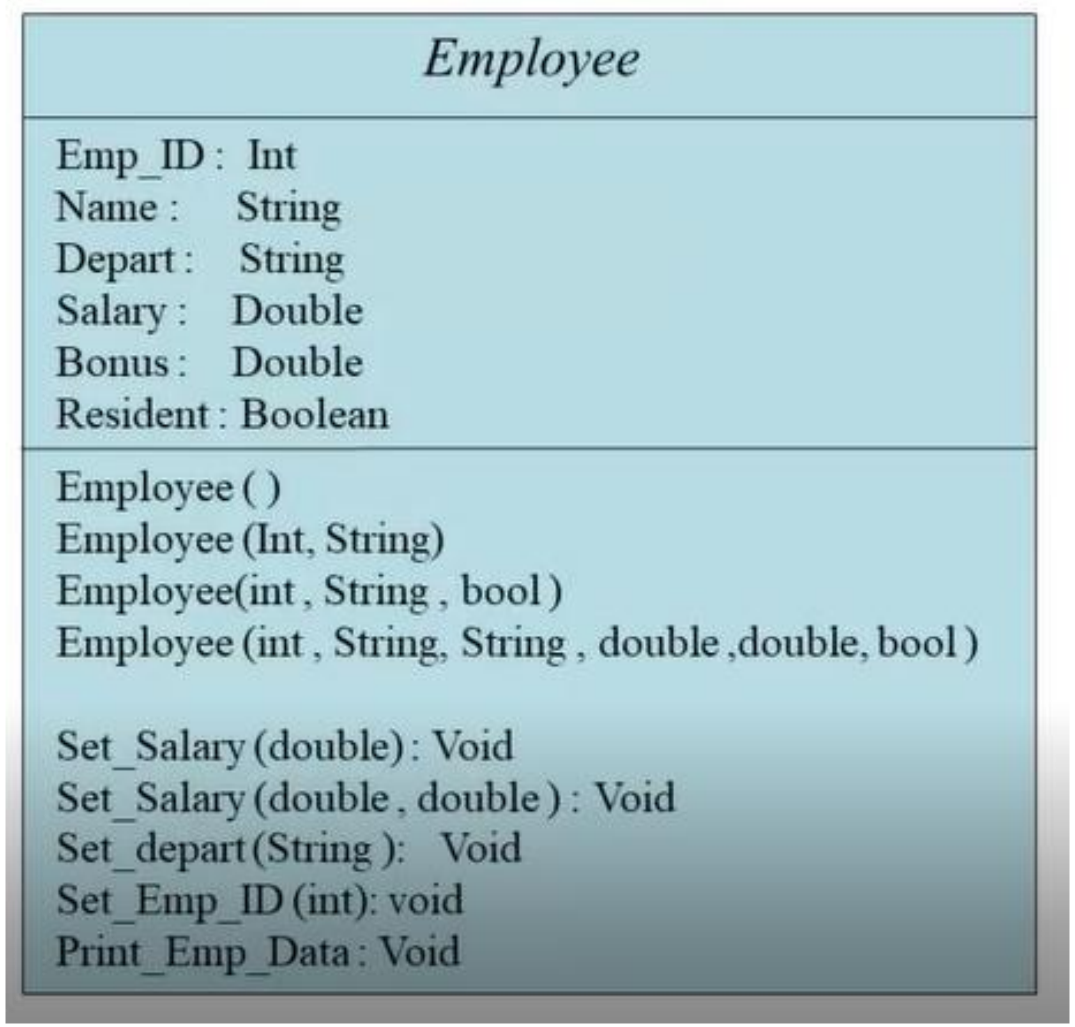
- Two or more methods in a class may have the same name as long as their signatures are different.
- Method signature (No of Args – Types of Args – Order of Args)
- When this occurs, it is called *method overloading*. This also applies to constructors.
- Method overloading is important because sometimes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

Example for overloading



Employee

Emp_ID : Int
Name : String
Depart : String
Salary : Double
Bonus : Double
Resident : Boolean

Employee ( )
Employee (Int, String)
Employee(int , String , bool)
Employee (int , String, String , double ,double, bool)

Set_Salary (double) : Void
Set_Salary (double , double) : Void
Set_depart (String): Void
Set_Emp_ID (int): void
Print_Emp_Data : Void

```
package javaapplication6;

public class Employee {
    int emp_id;
    String ename;
    String depart;
    double salary;
    double bonus;
    boolean resident;

    public Employee()
    {
        emp_id = 100;
        ename = "No Name";
        salary = 3000;
        bonus = 500;
        depart = "No Assigned Yet";
        resident = true;
    }
```

```
public Employee(int idno, String n)
{
    emp_id = idno;
    ename = n;
}

public Employee(int idno, String n, boolean r)
{
    //constructor chain
    this(idno, n);
    resident = r;
}
public Employee(int idno, String n, double s, double b, String d, boolean r)
{
    //constructor chain
    this(idno, n, r);
    depart = d;
    salary = s;
    resident = r;
}
```

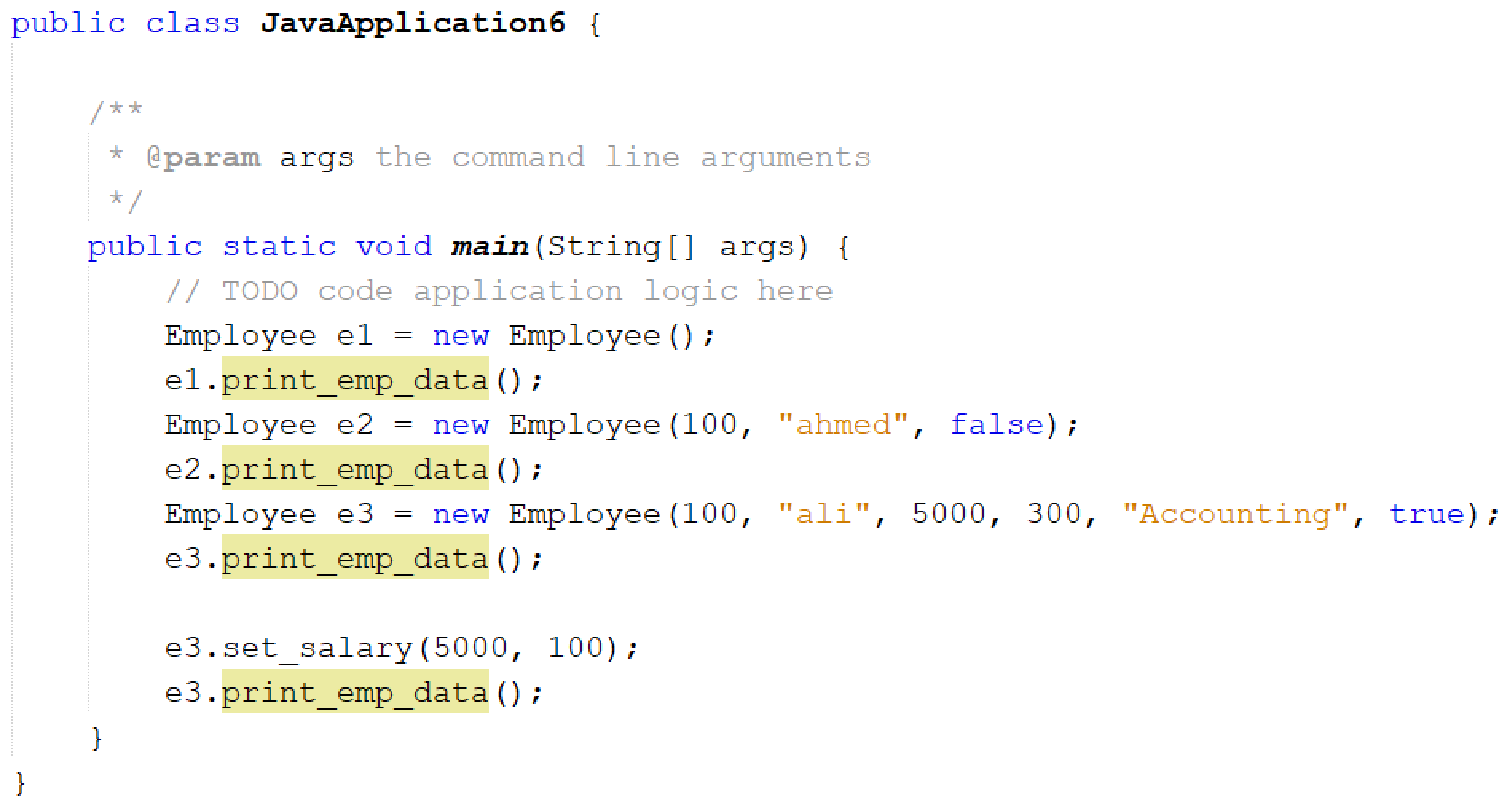
```
public void print_emp_data()
{
    System.out.println("ID "+emp_id);
    System.out.println("ename "+ename);
    System.out.println("depart "+depart);
    System.out.println("salary "+salary);
    System.out.println("bonus "+bonus);
    System.out.println("resident "+resident);


}

public void set_salary(double s)
{
    salary = s;
}

public void set_salary(double s, double b)
{
    this.set_salary(s);
    bonus = b;

}
```

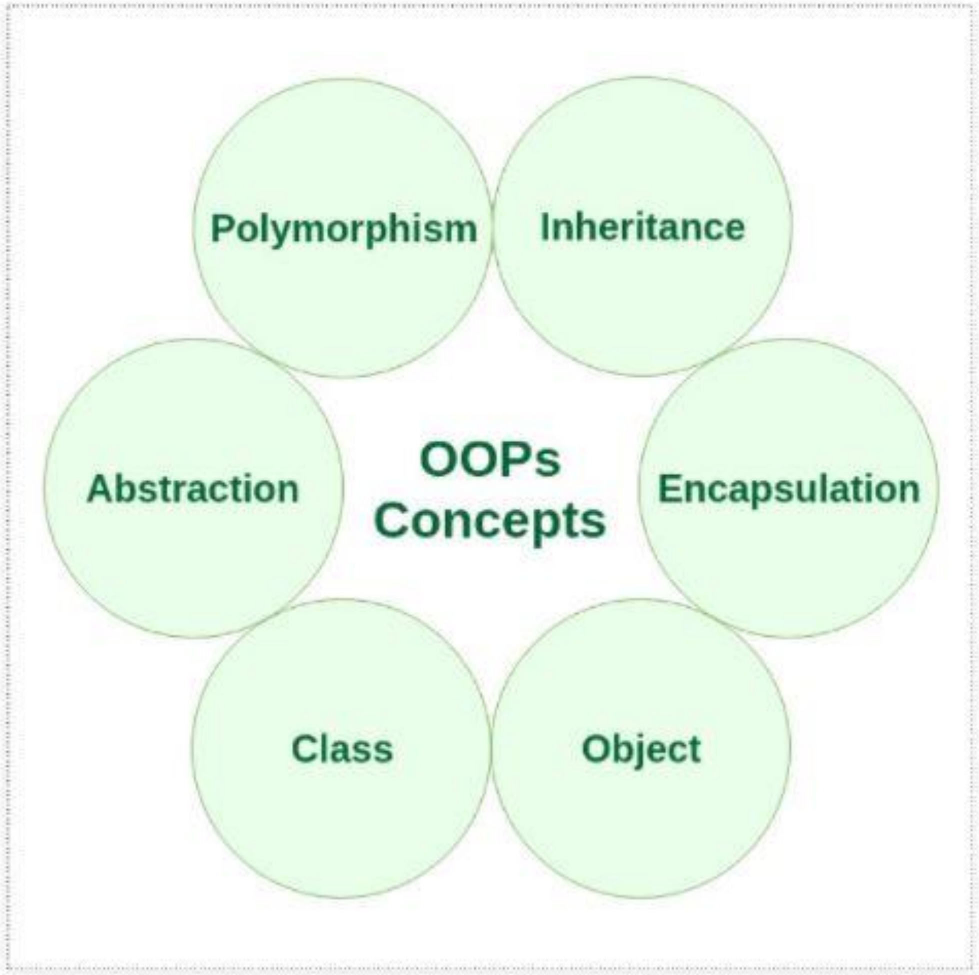


```java
public class JavaApplication6 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Employee e1 = new Employee();
        e1.print_emp_data();
        Employee e2 = new Employee(100, "ahmed", false);
        e2.print_emp_data();
        Employee e3 = new Employee(100, "ali", 5000, 300, "Accounting", true);
        e3.print_emp_data();

        e3.set_salary(5000, 100);
        e3.print_emp_data();
    }
}
```

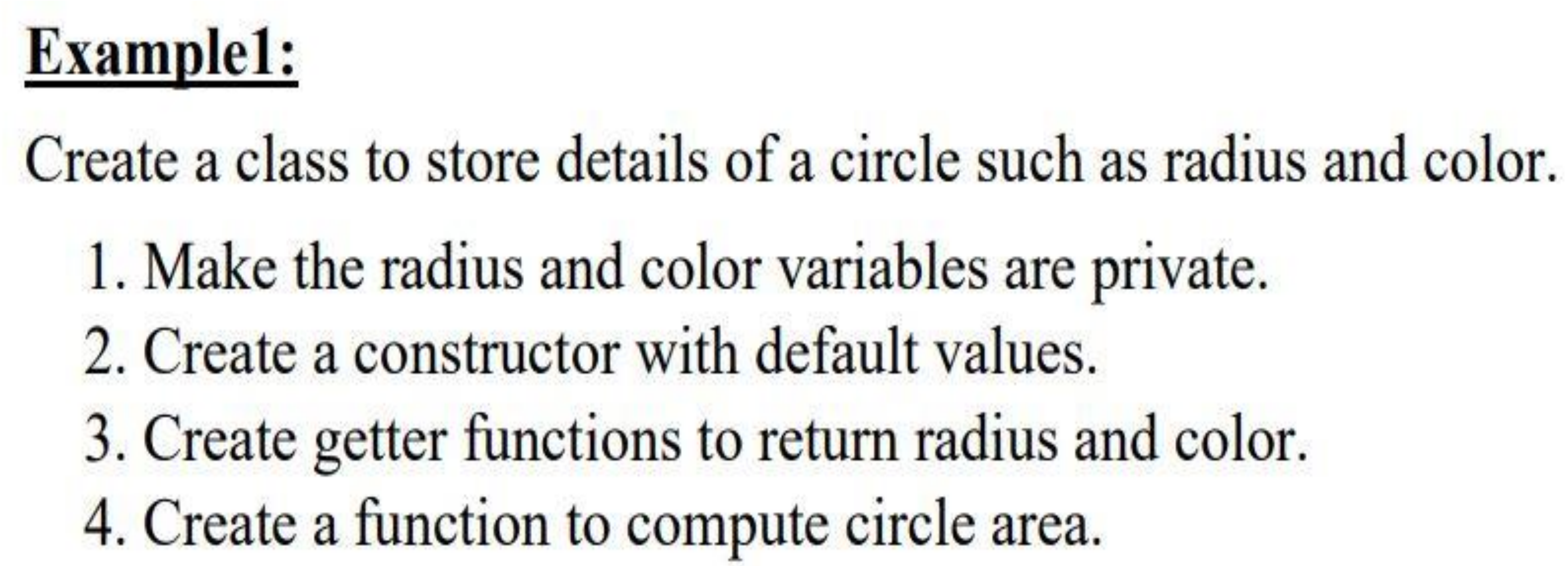
Characteristics of an Object-Oriented Programming language



Encapsulation

- The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users.
- To achieve this, you must declare class variables/attributes as private (cannot be accessed from outside the class).
- If you want others to read or modify the value of a private member, you can provide public **get** and **set** methods.

Assignment 1:

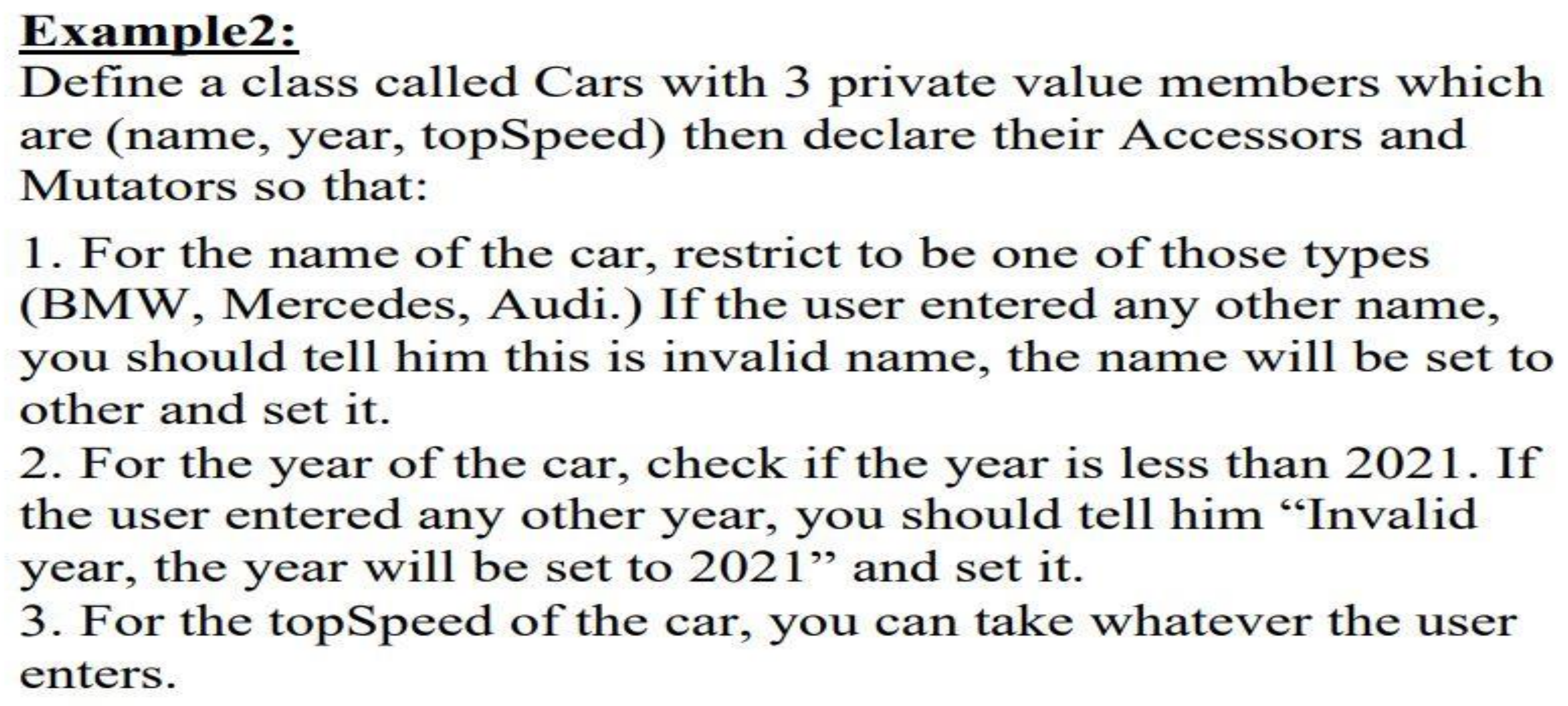


<u>Example1:</u>

Create a class to store details of a circle such as radius and color.

1. Make the radius and color variables are private.
2. Create a constructor with default values.
3. Create getter functions to return radius and color.
4. Create a function to compute circle area.

Assignment 2:

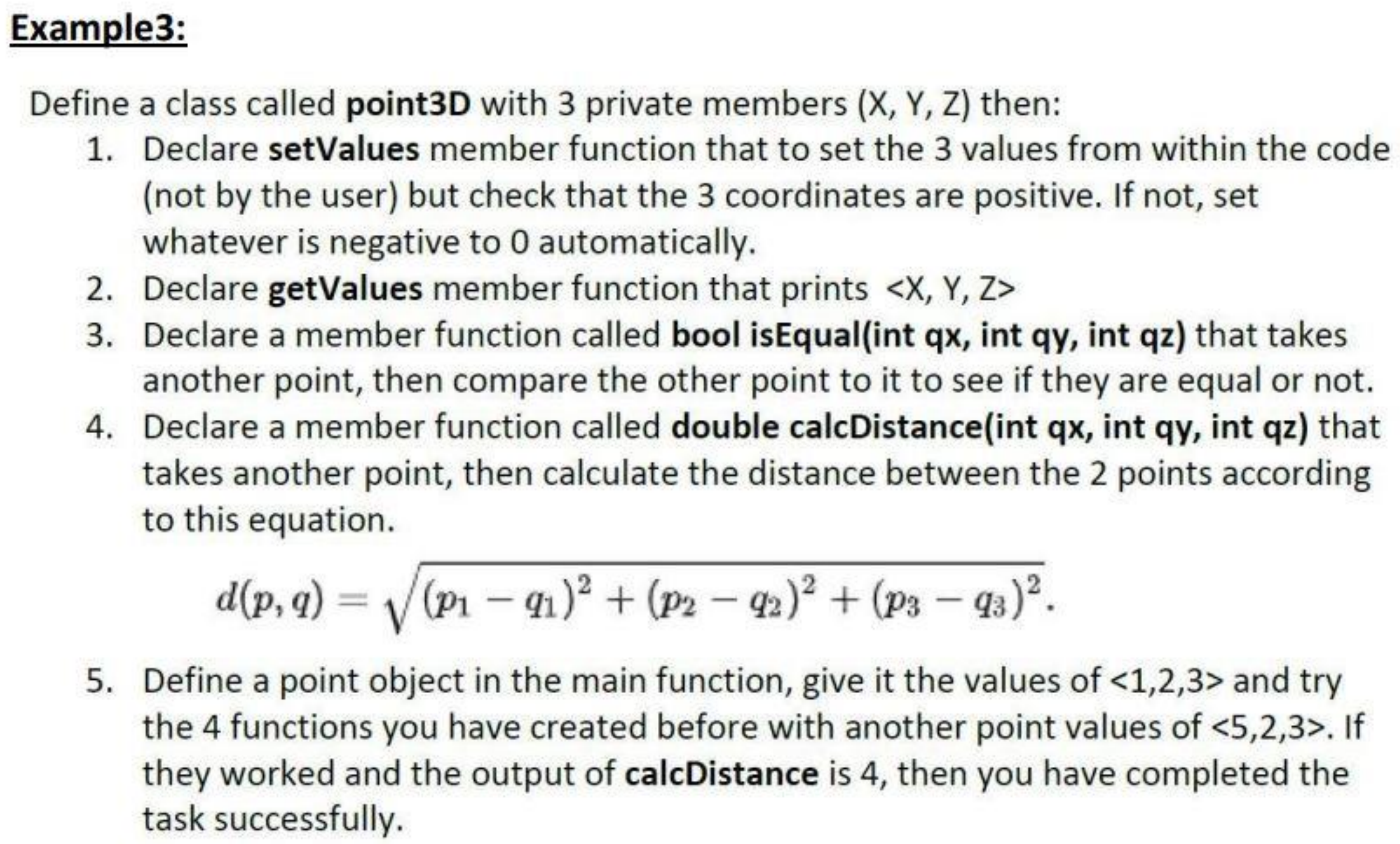


Example2:

Define a class called Cars with 3 private value members which are (name, year, topSpeed) then declare their Accessors and Mutators so that:

1. For the name of the car, restrict to be one of those types (BMW, Mercedes, Audi.) If the user entered any other name, you should tell him this is invalid name, the name will be set to other and set it.
2. For the year of the car, check if the year is less than 2021. If the user entered any other year, you should tell him “Invalid year, the year will be set to 2021” and set it.
3. For the topSpeed of the car, you can take whatever the user enters.

Assignment 3:



Example3:

Define a class called **point3D** with 3 private members (X, Y, Z) then:

1. Declare **setValues** member function that to set the 3 values from within the code (not by the user) but check that the 3 coordinates are positive. If not, set whatever is negative to 0 automatically.
2. Declare **getValues** member function that prints <X, Y, Z>
3. Declare a member function called **bool isEqual(int qx, int qy, int qz)** that takes another point, then compare the other point to it to see if they are equal or not.
4. Declare a member function called **double calcDistance(int qx, int qy, int qz)** that takes another point, then calculate the distance between the 2 points according to this equation.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}.$$

5. Define a point object in the main function, give it the values of <1,2,3> and try the 4 functions you have created before with another point values of <5,2,3>. If they worked and the output of **calcDistance** is 4, then you have completed the task successfully.

Thank you