Software Development and Best Practices

Part 1 : Object-oriented programming Revision

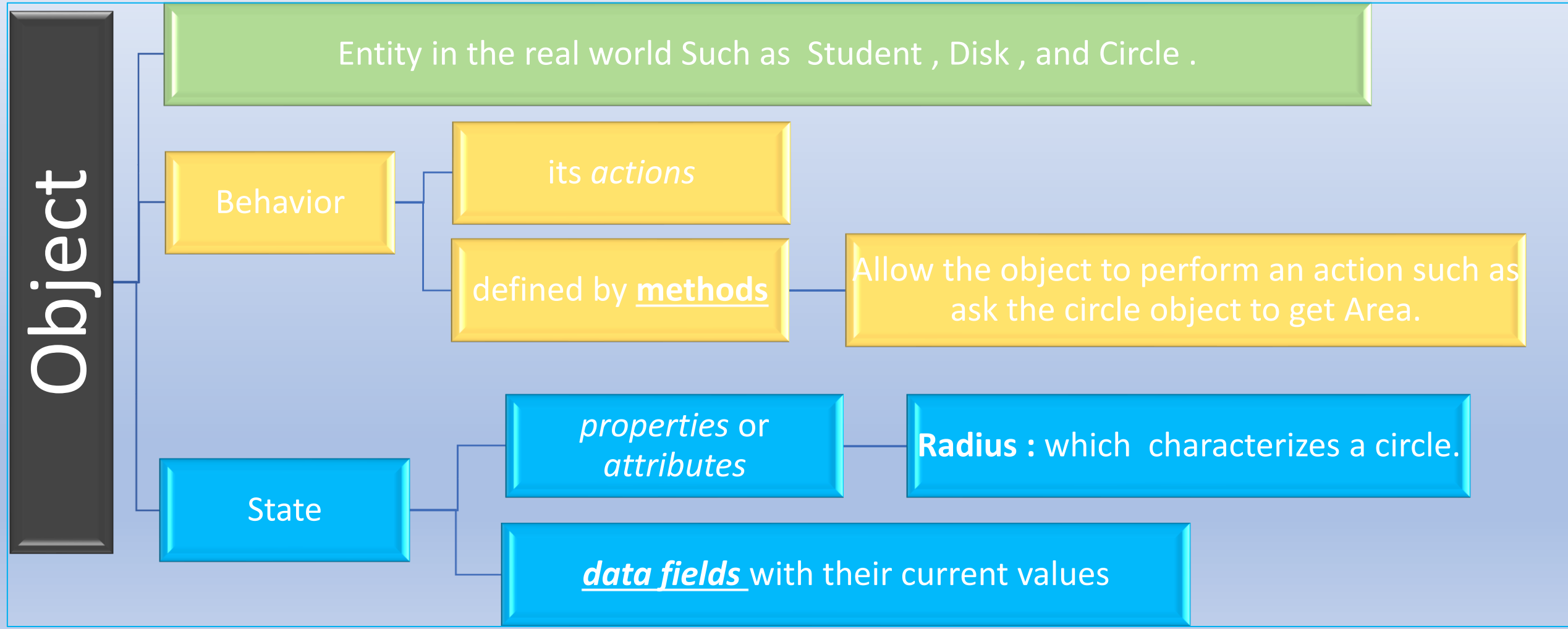
Mahmoud Khalaf Saeed

Section (1)

Objects and classes

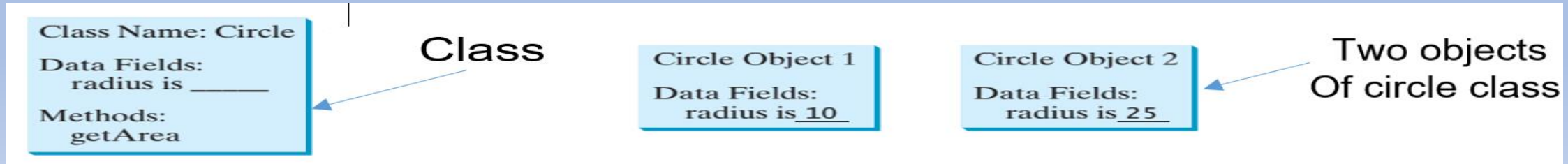
Object-oriented programming (OOP)

OOP : is the programming using objects



Class

- Objects of the same type are defined using a common class
- template, blueprint, or *contract that* defines data fields and methods
- ***instantiation*** — Creating an instance (object) of the class
- A Java class uses variables to define data fields and methods to define actions



constructor

- class provides methods of a special type, known as constructors
- designed to perform initializing actions, such as initializing the data fields of objects
- A constructor must have the same name as the class itself.
- Constructors do not have a return type—not even **void**.
- Constructors are invoked using the **new** operator when an object is created

constructor

The constructor is denoted as

```
ClassName(parameterName: parameterType)
```

constructors can be overloaded (i.e., multiple constructors can have the same name but different signatures)

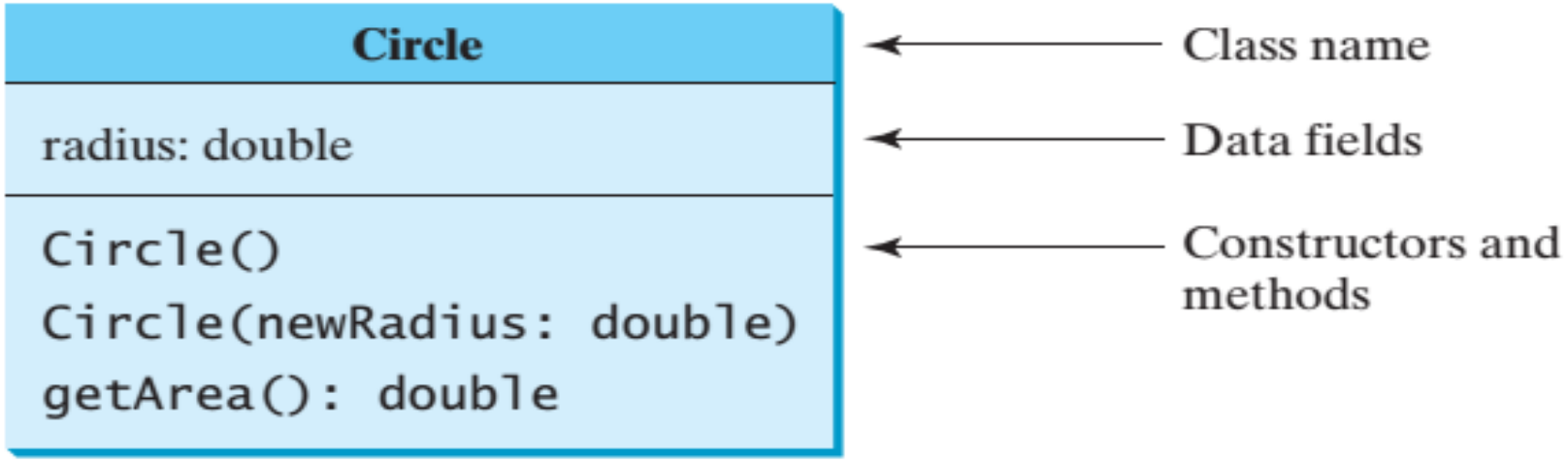
default constructor **(constructor with an empty body is implicitly defined), is provided automatically *only if no constructors are explicitly defined in the class.***

There are two types of constructor (*no-argument constructor. , argument constructor.*)

UML class diagram

```
dataFieldName: dataFieldType
```

UML Class Diagram



The method is denoted as

```
methodName(parameterName: parameterType): returnType
```

```java
package MyPackage;
public class Circle
{
    //data fields
    double radius ;

    // non parameterized constructor
    public Circle()
    {
        radius = 1.2 ;
    }
    // parameterized constructor
    public Circle( double userRadius)
    {
        radius = userRadius ;
    }
    // to get the area of circle
    public double getArea ()
    {
        return Math.PI * radius * radius ;
    }

    // to get the diameter of circle
    public double getDiameter()
    {
        return 2 * radius ;
    }
}
```

```java
package MyPackage;
public class MainClass
{
    public static void main(String[] args)
    {
        Circle circle1 = new Circle();
        System.out.println("the area of circle1 is " + circle1.getArea());
        System.out.println("the diameter of circle1 is " + circle1.getDiameter());

        System.out.println("--------------------------------------------------.");

        Circle circle2 = new Circle(1.8);
        System.out.println("the area of circle2 is " + circle2.getArea());
        System.out.println("the diameter of circle2 is " + circle2.getDiameter());

        System.out.println("--------------------------------------------------.");
        circle2.radius = 12 ;
        System.out.println("the area of circle2 after modification is " + circle2.getArea());
        System.out.println("the diameter of circle2 after modification is " + circle2.getDiameter());
    }

}
```

Expected Output

```
run:
the area of circle1 is 4.523893421169302
the diameter of circle1 is 2.4
-----------------------------------------------.
the area of circle2 is 10.17876019763093
the diameter of circle2 is 3.6
-----------------------------------------------.
the area of circle2 after modification is 452.3893421169302
the diameter of circle2 after modification is 24.0
```

DATA FIELDS DEFAULT VALUES

```java
public class Student
{
    int age ;
    String name ;
    boolean isScienceMajor;
    char gender;
    public static void main(String[] args)
    {
        Student std = new Student();

        System.out.println("default age is " + std.age);
        System.out.println("default name is " + std.name);
        System.out.println("default isScienceMajor is " + std.isScienceMajor);
        System.out.println("default gender is " + std.gender);
    }
}
```

```java
static char c = '\u0000';
```

```java
System.out.println(std.gender == c);
```

Expected Output

```
default age is 0
default name is null
default isScienceMajor is false
default gender is
```

Null character is printed for char data type which is '\u0000' in memory

The last print statement will print True .

Static Variables and Methods

- An instance variable is tied to a specific instance of the class; it is not shared among objects of the same class as radius in class circle
- If you want all the instances of a class to share data, use ***static variables***, also known as *class variables*.
- **Static variables** store values for the variables in a common memory location
- Java supports static methods as well as static variables.
- ***Static methods*** can be called without creating an instance of the class.

```java
public class Student
{
    static int no_of_Students;
    static int age ;
    public Student(int a)  // constructor used to updata no_of_Students every creation
    {
            no_of_Students ++ ;
            age = a ;
    }
    static  int get_no_of_Students ()
    {
        return no_of_Students ;
    }
    public static void main(String[] args)
    {
        Student std1 = new Student(21);
        Student std2 = new Student(99);
        System.out.println("no of students is " + get_no_of_Students()); // print 2
        //note that invoking static method directly not through object
        System.out.println("age of students is " + age); // print 99
    }

}
```

Note that : Final update of age is 99 so it will print the final update of the static variable as a result we use final in constant variables

NOTE THAT

- Static variables and methods can be used from instance or static methods in the class.
- However, instance variables and methods can be used only from instance methods, not from static methods
- Remember that static variables and methods don’t belong to a particular object

```java
package MyPackage;
public class Student
{
    static  int age ;
    int id ;
    static int getAge () {return age ; }

    int getAge2()
    {
        return age +50 ; // static variable used in an instance method
    }
    public static void main(String[] args)
    {
        Student std = new Student();
        int ag = age ;
    //int iid = id ;
    // wrong due to id is an instance variable invoked only through intances
        int iid = std.id ;
        System.out.println(iid);
    //System.out.println(getAge2());
    // wrong due to getAge2()is an instance method invoked only through intances
        System.out.println(getAge());
    }
}
```

VISIBILITY MODIFIERS

Access modifiers	Same class	Same package	Subclass	Other packages
Public	Y	Y	Y	Y
Protected	Y	Y	Y	N
Not Access modifiers	Y	Y	N	N
Private	Y	N	N	N

PUBLIC

You can use the **public** visibility modifier for classes, methods, and data fields to denote that they can be accessed from any other classes.

NO VISIBILITY MODIFIER

- the classes, methods, and data fields are accessible by any class in the same package. This is called *package-private package-access*.

private

- The **private** modifier makes methods and data fields accessible only from within its own class.

```java
package MyPackage;
public class C2
{
    public static void main(String[] args)
    {
        C1 f = new C1();
        System.out.println(" the value of X before update is " + f.x);
        f.x = 10 ;
        System.out.println(" the value of X after update is " + f.x);
        System.out.println("****************************************");

        System.out.println(" the value of y before update is " + f.y);
        f.y = 50 ;
        System.out.println(" the value of y after update is " + f.y);
        System.out.println("****************************************");

       //System.out.println(" the value of z before update is " + f.z);
        //you can not access the z data field due to it is a privte
        // in the class C1 [only class C1 can access it]
        // same thing for methods
    }

}
```

```java
package MyPackage;
public class C1
{
    public  int x ;
    int y ;
    private int z ;

    public void m1 ()
    {
        System.out.println("HI from method 1");
    }
    void m2 ()
    {
        System.out.println("HI from method 2");
    }
   private void m3 ()
    {
        System.out.println("HI from method 3");
    }
}
```

If a class is not defined public, it can be accessed only within the same package. As shown in Figure 8.14, C1 can be accessed from C2 but not from C3.

```
package p1;

class C1 {
  ...
}
```

```
package p1;

public class C2 {
  can access C1
}
```

```
package p2;

public class C3 {
  cannot access C1;
  can access C2;
}
```

A visibility modifier specifies how data fields and methods in a class can be accessed from outside the class. There is no restriction on accessing data fields and methods from inside the class.

DATA FIELD ENCAPSULATION

Encapsulation in Java is **a mechanism of wrapping the data (variables) and code acting on the data** (methods) together as a single unit.

In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class.

To prevent **direct modifications** of data fields, you should declare the data fields private, using the **private** modifier. This is known as *data field encapsulation*.

DATA FIELD ENCAPSULATION

A private data field cannot be accessed by an object from outside the class that defines the private field. But often a **client needs** to retrieve and modify a data field.

To make a private data field accessible, provide a ***get* method** to return its value. To enable a private data field to be updated, provide a ***set* method** to set a new value.

get method is referred to as a ***getter* (or *accessor*),** and a **set** method is referred to as a ***setter* (or *mutator*).**

BACK TO CIRCLE CLASS

The - sign indicates private modifier →

Circle	
-radius: double	The radius of this circle (default: 1.0).
-numberOfObjects: int	The number of circle objects created.
+Circle()	Constructs a default circle object.
+Circle(radius: double)	Constructs a circle object with the specified radius.
+getRadius(): double	Returns the radius of this circle.
+setRadius(radius: double): void	Sets a new radius for this circle.
+getNumberOfObjects(): int	Returns the number of circle objects created.
+getArea(): double	Returns the area of this circle.

Note : To prevent data from being tampered with and to make the class easy to maintain, declare data fields private.

```java
package MyPackage;
public class Circle
{
    //The radius of the circle
    private double radius = 1;
    private static int numberOfObjects = 0;
    /** Construct a circle with radius 1 */
    public Circle() { numberOfObjects++; }
    public Circle(double newRadius)
    {
        radius = newRadius;
        numberOfObjects++;}
// Return radius
    public double getRadius ()
    {
        return radius;}
// Set a new radius
    public void setRadius(double newRadius)
    {
        radius = (newRadius >= 0) ? newRadius : 0; }
//Return numberOfObjects */
    public static int getNumberOfObjects()
    {
        return numberOfObjects;
    }
    //Return the area of this circle */
    public double getArea()
    {
        return radius * radius * Math.PI;
    }
}
```

```java
package MyPackage;
public class TestCircle
{
    public static void main(String[] args)
    {
        Circle cir = new Circle( 0.5 );
        System.out.println("The area of the circle of radius " + cir.getRadius()
        + " is " + cir.getArea());

       cir.setRadius(cir.getRadius() * 3 );
       System.out.println("The area of the circle of radius " + cir.getRadius()
        + " is " + cir.getArea());
       System.out.println("The number of objects created is " + cir.getNumberOfObjects());
    }
}
```

```
run:
The area of the circle of radius 0.5 is 0.7853981633974483
The area of the circle of radius 1.5 is 7.0685834705770345
The number of objects created is 1
```

PASSING OBJECTS TO METHODS

```java
package MyPackage;
public class Circle2
{
    private double radius ;

    public Circle2(double newRadius)
    {
        radius = newRadius;
    }
    public double getRadius ()
    {
        return radius;
    }
    public double getArea()
    {
        return radius * radius * Math.PI;
    }
}
```

```java
package MyPackage;
public class TestCircle2
{
    public static void main(String[] args)
    {
        Circle2 myCircle = new Circle2( 0.8);
        printCircle(myCircle);
    }

    public static void printCircle (Circle2 cir)
    {
        System.out.println("the radius of the circle is " + cir.getRadius()
        + "\nand the area of the circle is " + cir.getArea() );
    }
}
```

Expected Output

```
the radius of the circle is 0.8
and the area of the circle is 2.0106192982974678
```

ARRAY OF OBJECTS

```java
package MyPackage;
public class TestCircle4
{    //to intialize an array of objects(circles)
    public static  Circle4[] createArrayOfCircle ()
    {
        Circle4[] arrOfCircles = new Circle4[5];
        for (int i = 0; i < 5; i++)
        {
            arrOfCircles [i] = new Circle4(Math.random() *100);
        }
        return arrOfCircles ;
    }
    // to print the circle [print radius , area]
    public static void printCircles (Circle4[] arrOfCircles)
    {
        for (int i = 0; i < arrOfCircles.length; i++)
        {
            System.out.println("the radius of circle  NO " + (i+1) + " = " + arrOfCircles[i].getRadius()
        + "and the area of circle NO " + (i+1)  + " = "  + arrOfCircles[i].getArea() );
        }
    }
    // to return the area of total circles created
    public static double getTotalArea (Circle4[] arrOfCircles)
    {
        double sum  =  0 ;
        for (int i = 0; i < arrOfCircles.length; i++)
        {
            sum +=  arrOfCircles[i].getArea();
        }
        return sum ;
    }
```

ARRAY OF OBJECTS

```java
// the entry point of the program
public static void main(String[] args)
{
    Circle4[] arrOfCircles = createArrayOfCircle (); //define arrOfCircles as an array
    printCircles(arrOfCircles);
    System.out.println("the total area of circles is " +getTotalArea(arrOfCircles));
}
```

Output similar to :

```
the radius of circle  NO 1 = 85.88140915751742and the area of circle NO 1 = 23171.182420084402
the radius of circle  NO 2 = 28.788429096479394and the area of circle NO 2 = 2603.6692098356357
the radius of circle  NO 3 = 2.2497272888153286and the area of circle NO 3 = 15.90045767889852
the radius of circle  NO 4 = 62.50477472392202and the area of circle NO 4 = 12273.721404406664
the radius of circle  NO 5 = 90.30582515113798and the area of circle NO 5 = 25620.134372826746
the total area of circles is 63684.607864832346
```

```java
package MyPackage;
public class Circle4
{
    public double radius ;

    public Circle4(double newRadius)
    {
        radius = newRadius;
    }
    public double getRadius ()
    {
        return radius;
    }
    public double getArea()
    {
        return radius * radius * Math.PI;
    }
}
```

Assignment

Implement the following TV class

TV	
`channel: int`	The current channel (1 to 120) of this TV.
`volumeLevel: int`	The current volume level (1 to 7) of this TV.
`on: boolean`	Indicates whether this TV is on/off.
`+TV()`	Constructs a default TV object.
`+turnOn(): void`	Turns on this TV.
`+turnOff(): void`	Turns off this TV.
`+setChannel(newChannel: int): void`	Sets a new channel for this TV.
`+setVolume(newVolumeLevel: int): void`	Sets a new volume level for this TV.
`+channelUp(): void`	Increases the channel number by 1.
`+channelDown(): void`	Decreases the channel number by 1.
`+volumeUp(): void`	Increases the volume level by 1.
`+volumeDown(): void`	Decreases the volume level by 1.

(*The* `Account` *class*) Design a class named `Account` that contains:

- A private `int` data field named `id` for the account (default `0`).
- A private `double` data field named `balance` for the account (default `0`).
- A private `double` data field named `annualInterestRate` that stores the current interest rate (default `0`). Assume all accounts have the same interest rate.
- A private `Date` data field named `dateCreated` that stores the date when the account was created.
- A no-arg constructor that creates a default account.
- A constructor that creates an account with the specified id and initial balance.
- The accessor and mutator methods for `id`, `balance`, and `annualInterestRate`.
- The accessor method for `dateCreated`.
- A method named `getMonthlyInterestRate()` that returns the monthly interest rate.
- A method named `withdraw` that withdraws a specified amount from the account.
- A method named `deposit` that deposits a specified amount to the account.

Draw the UML diagram for the class. Implement the class. Write a test program that creates an `Account` object with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the `withdraw` method to withdraw $2,500, use the `deposit` method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

Have a Good Day 😍