Object Oriented Programming

O·O·P

Chapter 9:-

Defining Classes and Objects:

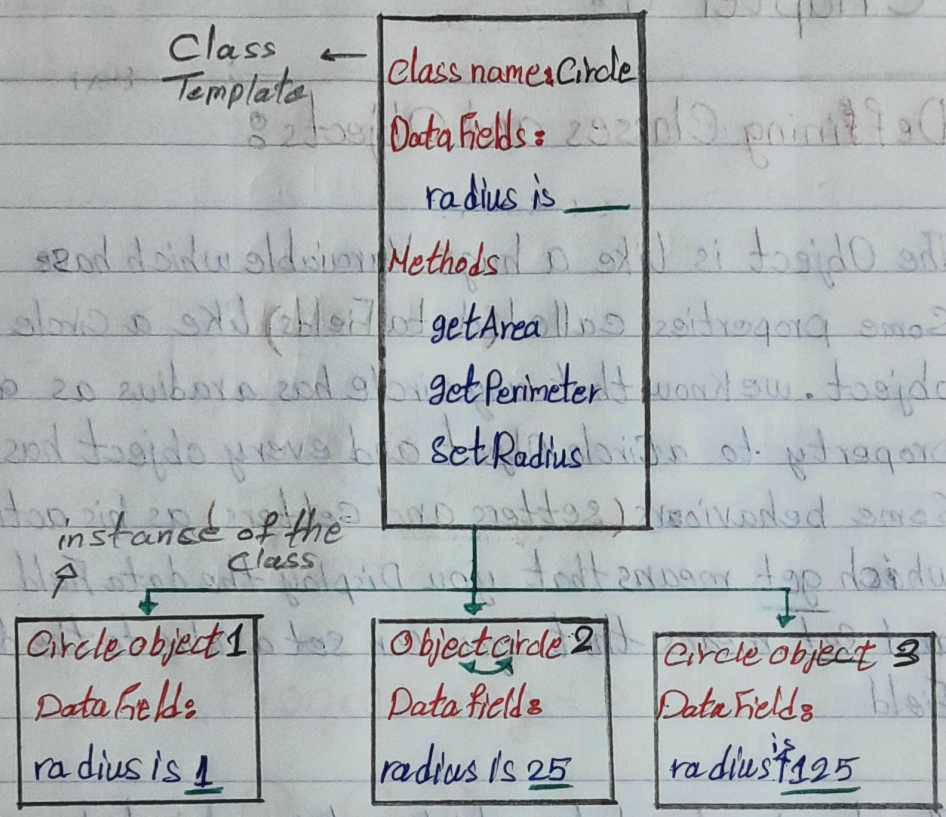
Def: The Object is like a huge Varaiable which hasse Some properties called (data Fields) like a Circle object. we know that any circle has a radius as a property to a Circle Object and every object has Some behaviours (setters and Getters) as his actions which get means that you Display the data Field and set means that you can set a value to the data Field

Def: The class is a tamplate which contains objects have The same data Fields and behaviors we called it also (blue Print) or (contract) it's Define what is data field and behavior of an object from it

Def: Instantiation is to creat an (Instance) object of a class

Def: Constructor is a special Type of methods which is invoked when you creat an Instance of a class automatically as a (Default Instance) or (Initializer)

ex:




```
Class Circle {
    double radius = 1;  -> Data Field
Circle(){
}
Circle(double newRadius) {
    radius = newRadius;
}
double getArea(){
    return Math.PI * Radius * Radius;
}
double getPerimeter(){
    return 2 * radius * Math.PI; }
```

Constructors

methods behaviors

```
doube setRadius(double newRadius) {
    radius = newRadius;          ---> method behavior
  }
} -> Final of circle class
```

notes: The constructors should be named as the name of the class and we can use overloading principle with it

note: The class cannot be run without define an instance of it but There is exception when the class is Abstract as the MathClass

Def: (UML) Unified Modeling Language is Koun as a class diagram which denotes to the data field as following written :

dataFieldName : dataFieldType

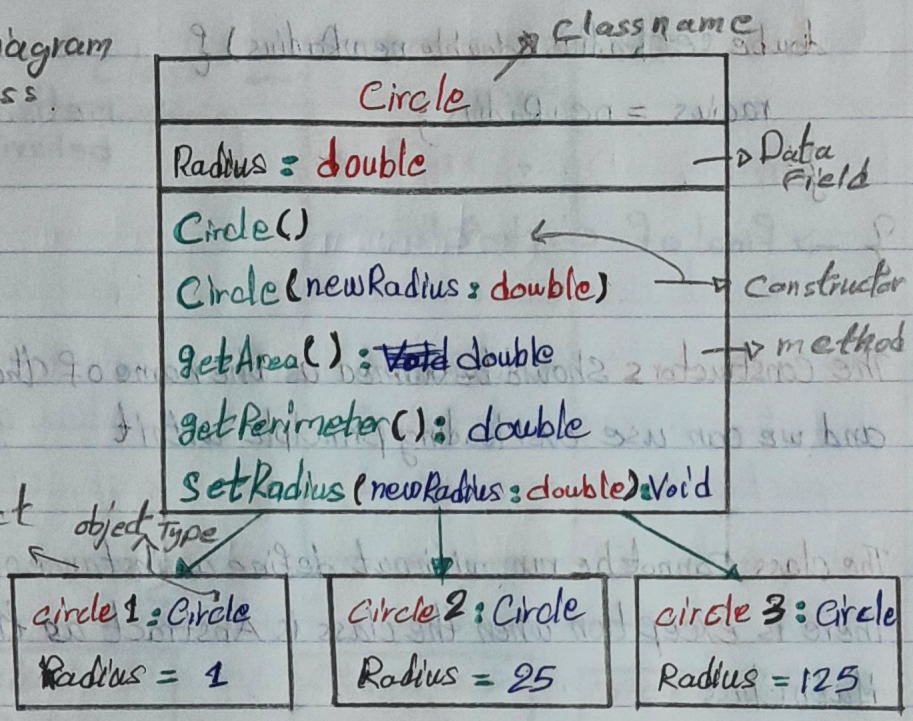
and denotes to the Constructor as following written :

ClassName(ParameterName : ParameterType)

and denotes to the method (as following written :

methodName(ParameterName : ParameterType) : ReturnType

ex: UML Diagram Class.



UML notation for objects

you can Creat an Instance of the circle class as following Syntax:

```
ReferenceVariable   : ok when you define a scanner Instance
      ↑
Circle C1 = new Circle();
            Reference type
```

and you can use the behaviors of the class as following Syntax:

```
dot operator
   ↑
C1.getArea();   → as when you use String class methods
↑
Instance method
```

note: The main class Called also Client of the class

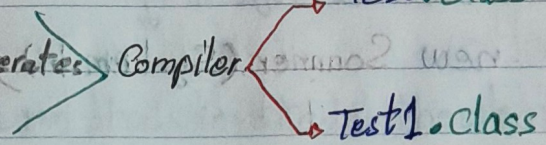
Public class

note: You can Put Two classes in one file but at most one is main

note: when the compiler Compiles a file contains Two or more Than Class, it's compile each class with a file by extention .class

ex:

```
public class Test {
}
Publ class Test1 {
}
```



note: you can combine more than 1 class in the main class as you write all data fileds and behaviors in it as methods

notes If you add + sign before any behavior That's mean Public modifier as 'publicstatic' and written as follows:

+ turn_on(): void

Constructing Objects Using Constructors:

you may be notice that the constructor has 3 characteristics (has the same name of the class) and (doesn't have a return data Type) and (invoked with new-operator) while you instantiation an instance of the class you can also make overloading on constructors as a new version of it

you can construct an object from a class as following Syntax:

```
new Classname(arguments);
```

(if it dosen't have arg's we called it no-arg Constructor)

as the Scanner Object ↓

```
new Scanner(System.in);
```

note: If you don't have any default Constructors (Constructor without arguments) it's created automatically with JDK

note: The Dot operator also in known as an object member access operator

you can access on the datafield of an object as following syntax:

```
objectRefVar.datafield;
```

(Instance Variable)

we have an exception in Recalling the methods in Math Class which you can invoke them as following Syntax:

```
Math.methodname();
```

Because this method is a static method (followed by static)

ex: this method is a static method:

```
public static double max(double n1, double n2)
```

static keyword

note: The static methods are methods invoked without an instance of the class but only by the class name as the Math class

you can creat or make initialization manually as the following Syntax:

```
new Circle();
```

and you can access on it's actions as following Syntax:

anonymous object

```
new Circle(35).getArea();
```

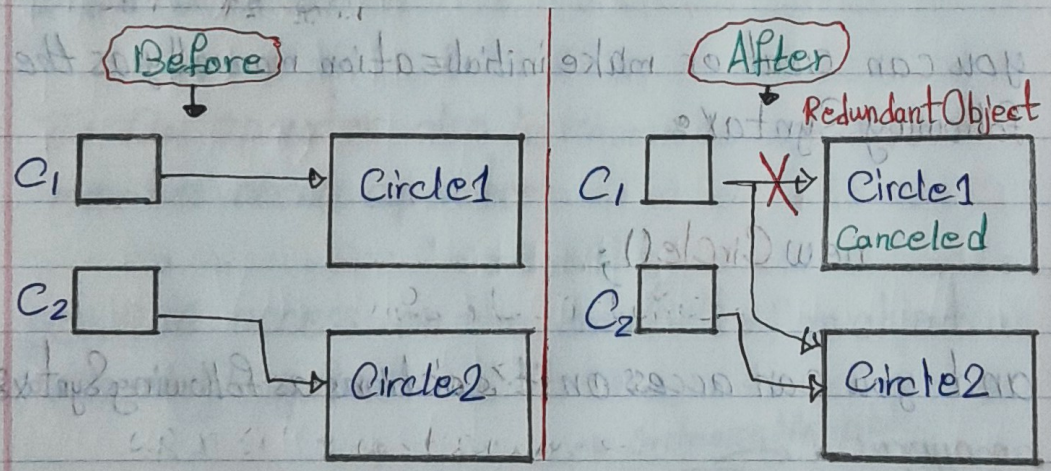
we have some default field values:

1. String → null
2. int → 0
3. boolean → false
4. Char → '\u0000'

note: NullPointerException is a Common RuntimeError it occurs when you invoke a method on a ReferenceVariable with a null Value

when you assign a Refrence ~~object~~ Variable to another the variables have the same reference after that and that is Kown as a (Garbage)

ex: Object Type assignment $C_1 = C_2$



now C1 Reference to circle2 object

when you make the preceeding assignment a process called (GarbageCollection) occurs which it ~~reclaims~~ (reclaims) the space Occupies by the Redundant Object or you can make it manually by assigning null to it's ReferenceVariable

Using Classes From the Java Library... 8

1 The Date Class :-

java.util.Date

+ Date() → default constructor
+ Date(elapseTime : long) Take Time in milliseconds since 1/1/1970 GMT
+ toString() : String returns Date and Time as a String
+ getTime() : long returns the number of Milliseconds since 1/1/1970 GMT
+ setTime(elapseTime : long) : void Take new Time

2 The Random Class :-

java.util.Random

+ Random() → Default Constructor
+ Random(seed : long) Random with specified seed ← Current Elapsed Time
+ nextInt() : int → returns Random int value
+ nextInt(n : int) : int returns Random int value between [0, n[
+ nextLong() : long returns Random long Value
+ nextDouble() : double returns Random double value [0, 1.0[
+ nextFloat() : float returns Random float value [0F, 1.0F[
+ nextBoolean() : boolean returns Random boolean Value

note: The seed of a random object is the number of indecies for the random number 487 has 3 seeds

note: If Two Random Objects have the same Seed They will generate identical sequences of numbers

[3] The Point2D Class:-

javafx.geometry.Point2D

+Point2D(x: double, y: double) Default Point (x,y)
+~~Point~~ distance(x: double, y: double): double
+distance(P: Point2D): double distance of 2 Points
+getX(): double returns x co-ordinate
+getY(): double returns y co-ordinate
+toString(): String returns String for Point

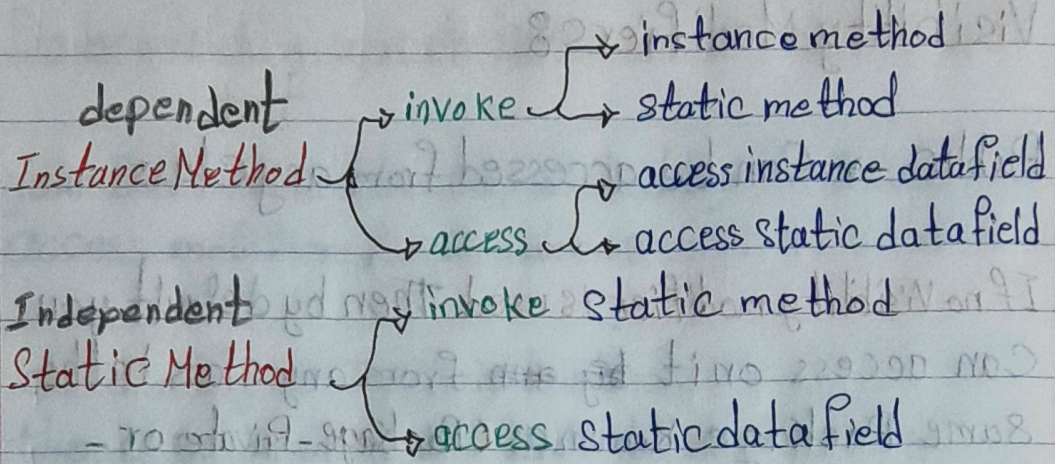
returns distance between this Point and (x,y) Point

Static Variables, Constants, and Methods

global

you can make a static Variables or Methods by The following Syntax:

```
static int a;
static int getNumber(){
}
Static final static int b;
```



note:

```
public class Test {
    int Count;
a:  public ? void main (String [] args){
        ——————
    }
b:  public ? int getCount() {
        return Count;
    }
c:  public ? int factorial (int n) {
        int result = 1;
        for (int i = 1; i <= n; i++)
            result *= i;
        return result;
    }
}
```

we will write Static Keyword with factorial and main method Because They aren't depend on instances

Visibility Modifiers:

1. Public → Can be accessed from any class

note: If no visibility modifier is used then by default you can access on it ~~by~~ ~~ano~~ from any class in the same Package known as (Package-Private or - Package access

Def: The Package is like a bag which contains classes and you can add it in the first line in the Program as following Syntax:

```
package packageName;
```

note: if the class is defined without Package statement it is said to be Placed in the default Package

2. Private → Can be accessed only from the own class

3. Protected → In the next Chapters

Data Field Encapsulation

your Data must be capsulated by the Private access modifier for 2 reasons:

1. data may be Tampered with
2. the class become difficult to maintain and Vulnerable to bugs

note: The setters also called Mutators and the getters also called accessors

ex:
```
public returnType getPropertyName()  ← getter
public boolean isPropertyName()  ← boolean accessor
public void set_propertyName(dataType propertyValue)
↳ setter
```

note: The minus sign '-' indicates a Private modifier

-Radius : double

The underlined 'text' indicates a static variable or method

numberofobjects(): int

Passing And Returning Objects from a Method :

you can Pass an object to a method by it's Reference with the object type as following Syntax :

```
Circle c1 = new Circle(5.0);
public static void printCircle(Circle c){
}
public static void main(String[] args){
    Circle c2 = new Circle(3.0);
    printCircle(c2);
}
```

Parameter object

Pass by Value

notes: The objects are Stored in a heap like arrays, Known that the array is an object

notes: Pass by Sharing Principle means that the object reference in the method is the same as the object being Passed

notes: when you make ~~Swapping~~ Swapping by References The content does not change in the objects you are Swapped

Array Of Objects:

you can creat an array of objects as following Syntax:

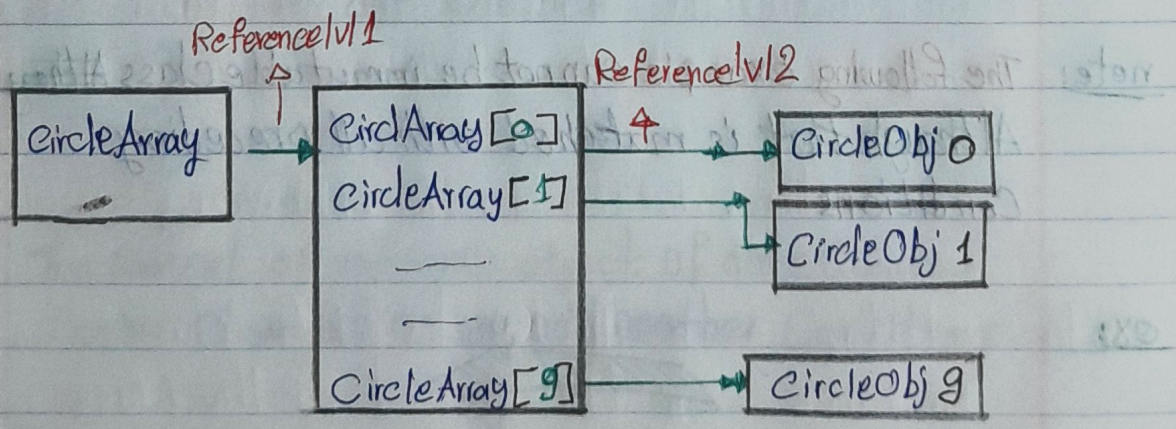
```
Circle[] circleArray = new Circle[10];
```

Class name — RefVar — Reference Type

This is An array of objects with 10 circle Type of object you can initialize it as following Syntax:

```
for(int i=0; i<circleArray.length; i++){
    CircleArray[i] = new Circle();
}
```

note: The Default Value of Object is null



Immutable Objects and Classes:

Def: Immutable means that a thing can not be change in classes that indicates to cancel the ability of access to the contents of objects and you can Creat immutable objects of immutable class String Class as an example is immutable Class (immutation) is to Delete the Setter Method from the class to make it immutable class) addition to that (make the data fields Private access)

note: Immutable class must have the following features:

1. all data fields with Private modifier
2. Does not contain any Setters

note: The following example can not be immutable class Although that is matched with all preceding conditions

ex:

```
public class Student {
    private int id;
    private String name;
    private java.util.Date dateCreated;
    public Student(int ssn, String newName) {
        id = ssn;
        name = newName;
        dateCreated = new java.util.Date();
    }
    public int getId(){
        return id;
    }
    public String getName(){
        return name;
    }
    public java.util.Date getDateCreated(){
        return dateCreated;
    }
}
```

*** The content of reference object of dateCreated can be changed so we add another condition:

[3] No accessor methods can return a reference to a data field that is mutable

The Scope of Variables:

Def: Class's variables and methods entire the class without Order and there is an Exception when we use a data field based on reference to another data field

note: If a local Variable has the same name as a ~~class's~~ Class's Variable the local variable takes Precedence and the class's variable with the same name is hidden

---

The this Reference

Def: The key word this refers to the object it self you can use it with data fields and behaviors which is instance somebody need it to reference hidden data fields or invoke an overloaded Constructor and you can right it as following Syntax:

```
this.datafield; this.behavior;
```

note: static → all objects of the class
Instance → this object of the class

note: this(arguments) → indecates to another Constructor
