Telegram link to ask for queries: https://t.me/+PIaEH8mE9sxkMzll

Q>

```
public class Test {
	public static void doSum(int x, int y){
		System.out.println("int sum is:: "+(x+y));
	}
	public static void doSum(Integer x, Integer y){
		System.out.println("Integer sum is:: "+(x+y));

	}
	public static void doSum(double x, double y){
		System.out.println("double sum is:: "+(x+y));
	}
	public static void doSum(float x, float y){
		System.out.println("float sum is:: "+(x+y));
	}
	public static void main(String[] args) {
		doSum(10,20);
		doSum(10.0,20.0);

	}
}
```

What is the result?

A. int sum is :: 30
float sum is :: 30.0

B. int sum is :: 30
double  sum is :: 30.0

C. Integer sum is :: 30
double sum is :: 30.0

D.  Integer sum is:: 30
float sum is :: 30.0

Answer: B

Q>

Consider the following snippet and predict the output

```
public class Test{
	public static void main(String... args){
		String language="java";

		while(language.equals("java"))
		{
			if(language.equals("java"))
				language=language.toUpperCase();
			if(language.equals("JAVA"))
				language=language.toLowerCase();

		}
		System.out.println(language); //line n1
	}
}
```

A. java
B. JAVA

C. Compile time error at line n1
D. Infintie time loop will run
E. None of the above

Answer: D

Q>
Consider below code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        int i = 0;
        for(System.out.print(i++); i < 2; System.out.print(i++)) {
            System.out.print(i);
        }
    }
}
```

What will be the result of compiling and executing Test class?
A. 112
B. 012
C. 011
D. 12
E. 01
F. CompilationError

Answer: C

Q>
Consider below code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        int i = 1;
        int j = 5;
        int k = 0;
        A: while(true) {
            i++;
            B: while(true) {
                j--;
                C: while(true) {
                    k += i + j;
                    if(i == j)
                        break A;
                    else if (i > j)
                        continue A;
                    else
                        continue B;
                }
            }
        }
        System.out.println(k);
    }
}
```

What will be the result of compiling and executing Test class?
A. Compilation Error
B. 6
C. 11
D. 15
E. Program never terminates it results in infinte loop
F. None of the above

Pillars of Object Orientation
++++++++++++++++++++++++++++
 1. Encapsulation
 2. inheritance
 3. Polymorphism


Datahiding => Our internal data should not go to outside world directly that is outside person can't access internal data directly.
              To promote datahiding, we need to use "access modifiers"
                  eg: private,protected
            It promotes security.

eg#1.
```
class Account
{
      private double balance;
}
```

Note: To access the balance variable data from the enduser, validation(collect username,password) will be performed by the application.


Abstraction
   Hiding internal implementation,but exposing the set of services is called "Abstraction".
   In java to bring abstraction, we use "abstract classes and interfaces"

eg: ATM GUI Screen
       BankPeople -> Highlight the set of services they are offering.
       EndUser    -> Using the offered services, the end user will use the application.


Encapsulation
 Binding of data and corresponding methods into single unit is called "Encapsulation".
 If any java class follows datahiding and abstraction such type of class is said to be "Encapsulated class".
      Encapsulation = Datahiding + abstraction.


eg#1.
```
//Encapsulated class
class TextBook
{
      //instance variable : encapuslated
      private int pages;

      //Setter method
      public void setPages(int pages)
      {
            if(pages > 0)
               this.pages = pages;
            else
               this.pages = 0;
      }
```

```
	//Getter method
	public int getPages()
	{
		return pages;
	}
}

class Test{
	public static void main(String[] args) {
		TextBook tb = new TextBook();
		tb.setPages(-100);
		int pageCount= tb.getPages();
		System.out.println("No of pages is :: "+pageCount);
	}
}
```

Syntax for Setter methods
++++++++++++++++++++++++
1. MethodName should be prefixed with set
2. It should be public
3. return type should be void
4. Compulsorily it should take an argument

```
	public void setXXXXX(XXXXX varaibleName)
	{
		this.variableName = variableName;
	}
```

Syntax for getter methods
++++++++++++++++++++++++
1. MethodName should be prefixed with get
2. It should be public
3. Return type should not be void
4. it is alwasy no argument method.

```
	public XXXXXX getXXXXX()
	{
		return variableName;
	}
```

Note: If the variable type is boolean, then for getter method the name as per the convention is "isVaraibleName()"

eg#1.

```
//Encapsulated class
class Doctor
{
	private String sname;
	private boolean married;

	public void setSname(String sname)
	{
		this.sname = sname;
	}
	public void setMarried(boolean married)
	{
		this.married =married;
	}
```

```
	public String getSname()
	{
		return sname;
	}
	public boolean isMarried()
	{
		return married;
	}
}
class Test{
	public static void main(String[] args) {
			Doctor d= new Doctor();

			d.setSname("karthik");
			d.setMarried(true);

			boolean status = d.isMarried();
			String name    = d.getSname();

			System.out.println("Name   is : " +name);
			System.out.println("Status is : " +status);
	}
}

Output
Name   is : karthik
Status is : true


Need of "this" keyword in java
++++++++++++++++++++++++++++
class Student
{
	String name;
	int age;
	float height;

	public void setData(String name, int age, float height)
	{
			name=name;
			age=age;
			height=height;
	}

	public void displayData()
	{
		System.out.println("Name   is : "+name);
		System.out.println("Age    is : "+age);
		System.out.println("Height is : "+height);
	}

}
class Test
{
	public static void main(String[] args)
	{
			Student std = new Student();
			std.setData("sachin",49,5.5f);
```

```
                  std.displayData();
      }
}
Output
Name   is : null
Age    is : 0
Height is : 0.0
```

As noticed in the above code, the method setData() would set the supplied value to the variables called name,age,height.
These supplied values are not been assigned to instance variables by jvm because "jvm by default in method will always" give
priority to "local varaibles" but not the instance variables. this problem is technically termed as "Shadowing".
To resolve this problem we use "this" keyword in java.

Solution using this keyword
++++++++++++++++++++++++++

```
class Student
{
      String name;
      int age;
      float height;

      public void setData(String name, int age, float height)
      {
            //we can use "this" to refer to Object
                  this.name=name;
                  this.age=age;
                  this.height=height;
      }

      public void displayData()
      {
            System.out.println("Name   is : "+this.name);
            System.out.println("Age    is : "+this.age);
            System.out.println("Height is : "+this.height);
      }

}
class Test
{
      public static void main(String[] args)
      {
                  Student std = new Student();
                  std.setData("sachin",49,5.5f);
                  std.displayData();
      }
}
Output
Name   is : sachin
Age    is : 49
Height is : 5.5
```

Dependancy Injection
++++++++++++++++++++

=> It refers to process of injecting the values to the instance variables of a class.

=> we can perform depedancy injection in 2 ways
a. through setter method.
b. through constructor

Dependancy injection using setter method
++++++++++++++++++++++++++++++++++++++
eg#1.

```
class Student
{
	//instance variables
	private String name;
	private int age;
	private float height;

	//setter methods
	public void setName(String name){
		this.name = name;
	}
	public void setAge(int age){
		this.age = age;
	}
	public void setHeight(float height){
		this.height = height;
	}

	//getter methods
	public String getName(){
		return name;
	}
	public int getAge(){
		return age;
	}
	public float getHeight(){
		return height;
	}
}
class Test
{
	public static void main(String[] args)
	{
			//Constructing the object
			Student std = new Student();

			//setting the values for instance variable
			std.setName("sachin");
			std.setAge(49);
			std.setHeight(5.5f);

			//getting the values from instance variables
			System.out.println("Name   is :: "+std.getName());
			System.out.println("Age    is :: "+std.getAge());
			System.out.println("Height is :: "+std.getHeight());
	}
}
```

Output

```
Name   is :: sachin
Age    is :: 49
Height is :: 5.5
```