

1. What is class & object?

According to OOPS each & everything is treated as one object

OBJECT is an entity which having its own State & behaviour

Where state represent Characteristics & Behaviour Represent Functionality

CLASS is an area where Programmer can declare state & Behaviour for an Object

OBJECT is replica of a class which fetching state & Behaviour from a class

2. Explain class & object , give real time example explain?

CLASS: a class is a group of objects which are common to all objects of one type

EXAMPLE: Let us consider two objects MI phone and iPhone. Both phone have some properties like width, height, OS, price and Actions are call (), send Message (), browser (), share ().

Both objects have some different properties and actions but the type is the same "Phone". This is the class. i.e. the name of the class is "Phone".

: An entity that has state and behaviour is known as an object. Here, State represents the properties and Behaviour represents the actions or functionality

EXAMPLE: Let's take an example of PHONE

Properties: like width, height, OS, price

Actions: are call (), send Message (), browser (), share ().

3. What is constructor, what is real use of constructor?

Constructor is a special member of a class

Constructor are mainly made for object creation as well as data member initialization

According to java rule each & every class should contain constructor, if programmer failed to provide a constructor then compiler will provide a constructor into the class.

Types of constructor

1)Default constructor: the constructor provided by compiler is known as default constructor

When programmer not provide the constructor compiler will provide the constructor into the class.

EX: Structure of default constructor

```
Class a{
    Test()
    {super(); //Default Constructor
    }
    Public static void main(String [] args)
    {
        Test t =new Test ();
    }
}
```

2)User declared Constructor:

In java programme programmer can provide his own constructor.

If programmer is providing a constructor then compiler will not provide a constructor.

```
Class a{
    Test()
    {super();
    }
    Public static void main(String [] args)
    {
        Test t =new Test ();
    }
}}
```

Default Constructor Are also 2 types.

a) Argument Constructor or Parametrize Constructor

Constructor with arguments(or you can say parameters) is known as Parameterized constructor.

```
class Demo
{   public Demo(int x)
    {
        System.out.println("This is a no argument constructor");
    }
    public static void main(String args[]) {
        new Demo(10);
    }
}
```

b) Non-argument Constructor or De-parameterize Constructor

Constructor with no arguments is known as **no-arg constructor**. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.

```
class Demo
{   public Demo()
    {
        System.out.println("This is a no argument constructor");
    }
    public static void main(String args[]) {
        new Demo();
    }
}
```

Real Use Of Constructor

To identify and inherit the properties of a superclass and to create an object by which we can use the features of the superclass

4. What is constructor overloading & why its required?

Declaring a constructor multiple times in a same class with argument this process is known as constructor overloading.

//Java program to overload constructors

```
class Student5{
    int id;
    String name;
    int age;
    //creating two arg constructor
    Student5(int i,String n){
        id = i;
        name = n; }
    //creating three arg constructor
    Student5(int i,String n,int a){
        id = i;
        name = n;
        age=a; }
    void display(){
        System.out.println(id+" "+name+" "+age);}
    public static void main(String args[]){
        Student5 s1 = new Student5(111,"Karan");
        Student5 s2 = new Student5(222,"Aryan",25);
        s1.display();
        s2.display(); } }
```

5. What is constructor chaining?

Constructor chaining is the process of calling one constructor from another constructor with respect to current object.

Constructor chaining can be done in two ways:

- Within same class: It can be done using `this()` keyword for constructors in same class
 - With Different class: by using `super()` keyword to call constructor from the base class.
- Constructor chaining occurs through inheritance.

6. What is the use of this & super keyword?

use of super keyword

1. To access the data members of parent class when both parent and child class have member with same name
2. To explicitly call the no-arg and parameterized constructor of parent class
3. To access the method of parent class when child class has overridden that method.

Use of this keyword

1. this can be used to refer current class instance variable.
2. this can be used to invoke current class method (implicitly)
3. this can be used to invoke current class constructor.
4. this can be passed as an argument in the method call.
5. this can be passed as argument in the constructor call.
6. this can be used to return the current class instance from the method.

7. What is the use of this() & super() statement

THIS() : Use to access another constructor in a same class.

It is used in only Constructor.

It should be the first statement of a constructor

It can be used only once

SUPER() : Use to access another constructor of a different class

It is used only in constructor.

It should be the first statement of a constructor.

It can be used only once.

8. How many ways, we can create an Object to Class ?

1. Using new Keyword

```
Employee emp1 = new Employee();
```

2. Using newInstance() Method Class class

```
Employee emp2 = Employee.class.newInstance();
```

3. Using Clone Method()

```
Employee emp4 = (Employee) emp3.clone();
```

4. Using new Instance() method of Constructor class

```
Constructor<Employee> constructor = Employee.class.getConstructor();
```

```
Employee emp3 = constructor.newInstance();
```

5. Using Deserialization

```
ObjectInputStream in = new ObjectInputStream(new FileInputStream("data.obj")); Employee emp5 = (Employee) in.readObject();
```

9. Explain the members of class

There are FIVE members in a class.

1. Member Variables (States)
2. Methods (Behaviours)
3. Constructor
4. Blocks (Instance/Static Blocks)
5. Inner Classes.

10. Difference between static and non-static variable

STATIC VARIABLE	NON STATIC VARIABLE
Static variables can be accessed using class name	Non static variables can be accessed using instance of a class
Static variables can be accessed by static and non static methods	Non static variables cannot be accessed inside a static method.
Static variables reduce the amount of memory used by a program.	Non static variables do not reduce the amount of memory used by a program
Static variables are shared among all instances of a class.	Non static variables are specific to that instance of a class.
Static variable is like a global variable and is available to all methods.	Non static variable is like a local variable and they can be accessed through only instance of a class.

11. What is the use of class loader & garbage collector in JAVA

Class Loader: The main work of class is to load .class file into JVM Memory. Class loader divides the work in three stages.

1. Loading
2. Linking
3. Initialization

Garbage Collector: In java, garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the De-Refer objects.

```
System.gc();
Runtime.getRuntime.gc();
```

12. Can we have two main method in One JAVA class?

Yes, you can have as many main methods as you like. You can have main methods with different signatures from `main(String[])` which Yes, you can have as many main methods as you like. You can

have main methods with different signatures from `main(String[])` which is called overloading, and the JVM will ignore those main methods.

You can have one public static void `main(String[] args)` method in each class. Some people use those methods for testing. They can test the operation of each class individually. The JVM will only invoke the public static void `main(String[] args)` method in the class you name when you write `java MyClass`.

```
public class TwoMain {  
    public static void main(String args1[])  
    {  
        System.out.println("First main");  
    }  
    public static void main(String args2[])  
    {  
        System.out.println("Second main");  
    }  
}
```

Those two methods have the same signature. The only way to have two main methods is by having two different classes each with one main method. The name of the class you use to invoke the JVM (e.g. `java Class1`, `java Class2`) determines which main method is called. This is called overloading, and the JVM will ignore those main methods.

13. Can we declare constructor as private? If yes Why?

The **use of private constructor** is to serve singleton classes. A singleton class is one which limits the number of objects creation to one. Using private constructor we can ensure that no more than one object can be created at a time. By providing a private constructor you prevent class instances from being created in any place other than this very class. We will see in the below example how to use private constructor for limiting the number of objects for a singleton class.

14. Can we declare Class as private?

We cannot declare top level class as private. Java allows only public and default modifier for top level classes in java. Inner classes can be private.

It is because, java follows oops concepts. If you make any class as private, it will not be accessible from another class. So there can't be inheritance, runtime polymorphism, abstraction etc.

But, nested class can be private. For instance:

```
PrivateClass.java
1 package programme;
2
3 class PrivateClass{
4     private class B{
5     void msg()
6     {
7         System.out.println("hello");
8     }
9 }
10 public static void main(String args[])
11 {
12     PrivateClass.B b=new PrivateClass().new B();
13     b.msg();
14 }
15 }
16
17
18
19
```

Problems Javadoc Declaration Console

<terminated> PrivateClass [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe
hello

15. Can we declare Class as static?

A static class i.e. created inside a class is called static nested class in java. It cannot access non-static data members and methods. It can be accessed by outer class name.

- It can access static data members of outer class including private.
- Static nested class cannot access non-static (instance) data member or method.

The screenshot shows an IDE with two tabs: 'PrivateClass.java' and 'StaticClass.java'. The 'StaticClass.java' tab is active, displaying the following code:

```
1 package programme;
2
3 public class StaticClass {
4
5     static int data=30;
6     static class Inner{
7         void msg(){System.out.println("data is "+data);}
8     }
9     public static void main(String args[]){
10         StaticClass.Inner obj=new StaticClass.Inner();
11         obj.msg();
12     }
13 }
14
15
16
```

Below the code editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> StaticClass [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (22-Sep-2019, 2:43:0)
data is 30
```

109. W.A.P PALINDROME?

```

1 package programme;
2
3 import java.util.Scanner;
4
5 public class PalindromeNumber {
6     public static void main(String[] args) {
7         Scanner scan=new Scanner(System.in);
8         System.out.println("Enter value of x");
9         int x=scan.nextInt();
10        int temp=x;
11        int sum=0;
12        int r=0;
13        while(x>0)
14        {
15            r=x%10;
16            sum=(sum*10)+r;
17            x=x/10;
18        }
19        if(temp==sum)
20        {
21            System.out.println("its a palindrom number :" + sum);
22        }
23        else
24        {
25            System.out.println("Its not palindrome number : " + sum);
26        }
27    }
28 }

```

Problems @ Javadoc Declaration Console

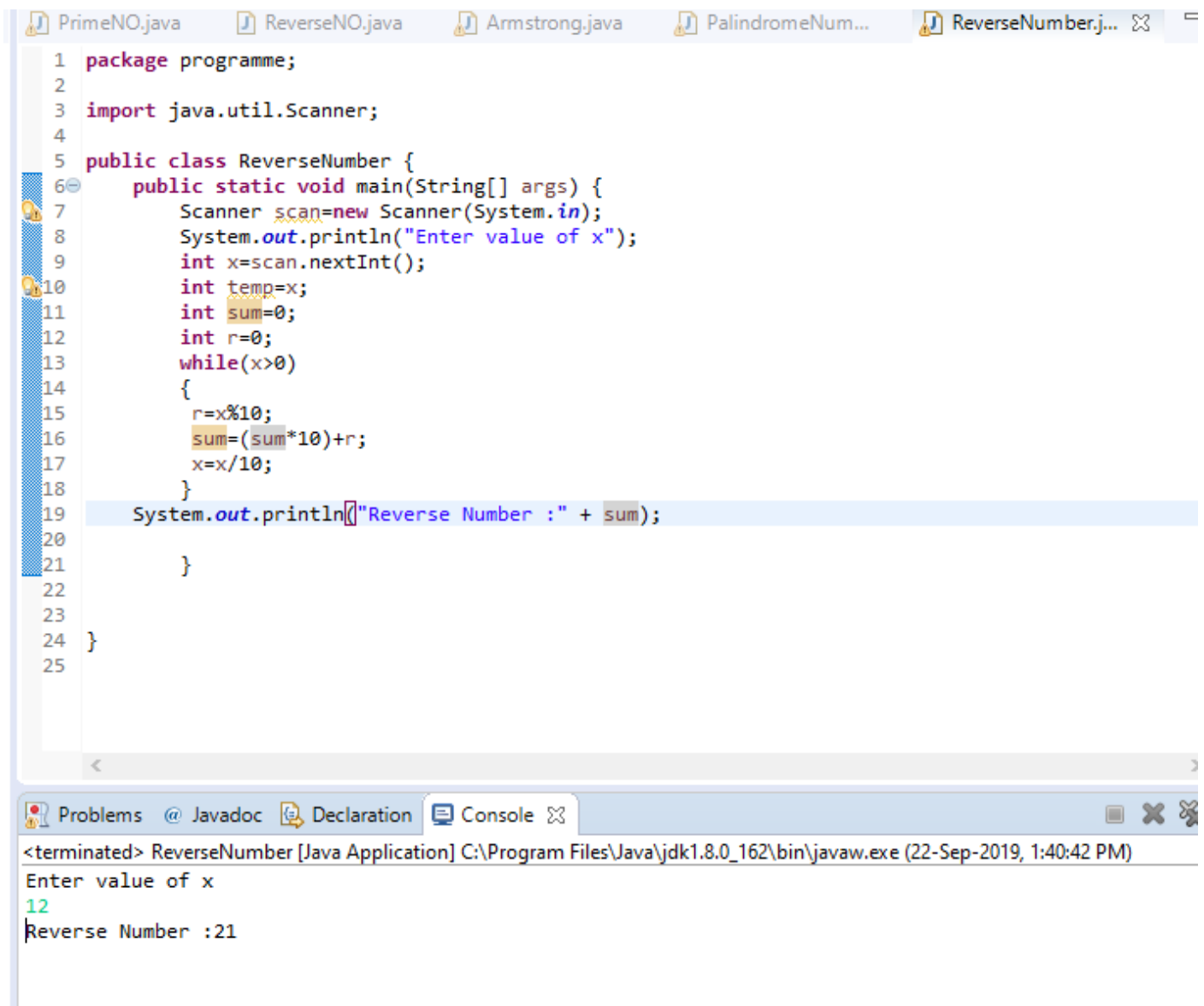
<terminated> PalindromeNumber [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe

Enter value of x

434

its a palindrom number :434

110. w.a.p for reverse number?



The screenshot shows an IDE with several open files: PrimeNO.java, ReverseNO.java, Armstrong.java, PalindromeNum..., and ReverseNumber.j... The code in ReverseNumber.java is as follows:

```
1 package programme;
2
3 import java.util.Scanner;
4
5 public class ReverseNumber {
6     public static void main(String[] args) {
7         Scanner scan=new Scanner(System.in);
8         System.out.println("Enter value of x");
9         int x=scan.nextInt();
10        int temp=x;
11        int sum=0;
12        int r=0;
13        while(x>0)
14        {
15            r=x%10;
16            sum=(sum*10)+r;
17            x=x/10;
18        }
19        System.out.println("Reverse Number : " + sum);
20
21    }
22
23 }
24 }
25 }
```

The console output shows the program execution:

```
<terminated> ReverseNumber [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (22-Sep-2019, 1:40:42 PM)
Enter value of x
12
Reverse Number :21
```

111. Armstrong number?

```
2
3 import java.util.Scanner;
4
5 public class Armstrong {
6     public static void main(String[] args) {
7         Scanner scan=new Scanner(System.in);
8         System.out.println("Enter value of x");
9         int x=scan.nextInt();
10        int temp=x;
11        int sum=0;
12        int r=0;
13        while(x>0)
14        {
15            r=x%10;
16            sum=(sum)+r*r*r;
17            x=x/10;
18        }
19        if(sum==temp)
20        {
21            System.out.println("No is Armstrong : " + sum);
22        }
23        else
24        {
25            System.out.println("No is not Armstrong : " + sum);
26        }
27    }
28 }
```

Problems @ Javadoc Declaration Console

<terminated> Armstrong [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (22-Sep-2019, 1:25:42 PM)

Enter value of x

56

No is not Armstrong : 341

112. Prime number or not?

```
PrimeNO.java
2
3 import java.util.Scanner;
4
5 public class PrimeNO{
6
7     public static void main(String[] args) {
8         Scanner scan=new Scanner(System.in);
9         System.out.println("Enter the value of num");
10        int num = scan.nextInt();
11        boolean flag = false;
12        for(int i = 2; i <= num/2; ++i)
13        {
14            // condition for nonprime number
15            if(num % i == 0)
16            {
17                flag = true;
18                break;
19            }
20        }
21        if (!flag)
22            System.out.println(num + " is a prime number.");
23        else
24            System.out.println(num + " is not a prime number.");
25    }
26 }
27
28
29
```

Problems @ Javadoc Declaration Console

<terminated> PrimeNO [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe

Enter the value of num

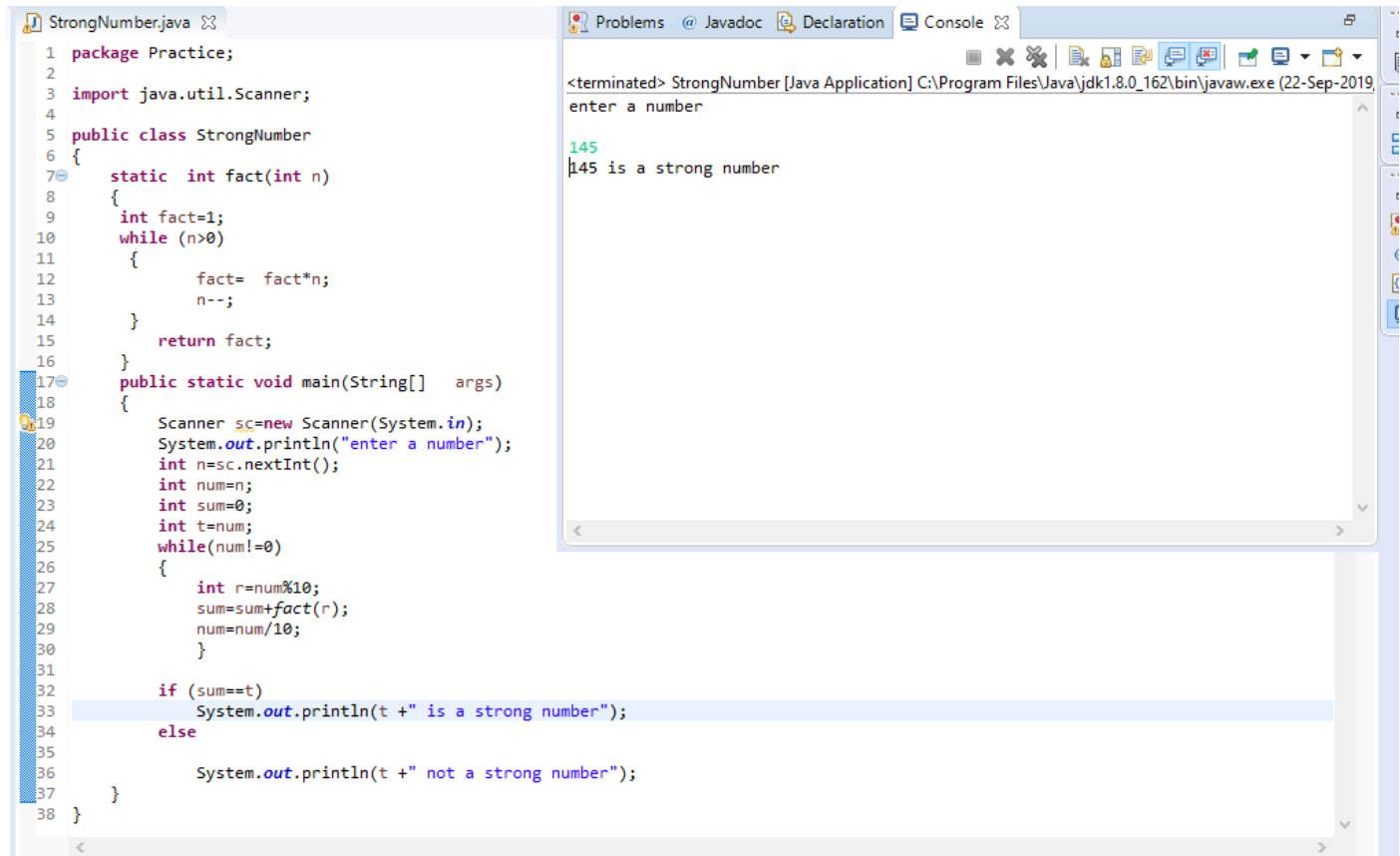
4

4 is not a prime number.

113. w.a.p to generate prime numbers between 1 to 1000?

```
PrimeNumber.java ✕
1 package Practice;
2
3 import java.util.Scanner;
4
5 public class PrimeNumber {
6
7     public static void main(String[] args)
8     {
9         Scanner scan=new Scanner (System.in);
10        System.out.println ("enter number");
11        int n=scan.nextInt();
12        System.out.println ("Prime numbers between 1 and " + n);
13        //loop through the numbers one by one
14        for (int i=1;i<n;i++)
15        {
16            boolean isPrime =true;
17            //check to see if the number is prime
18            for (int j=2;j<i;j++)
19            {
20                if (i%j==0)
21                {
22                    isPrime =false;
23                    break;
24                }
25            }
26            // print the number
27            if (isPrime)
28                System.out.println(i + " ");
29        }
30    }
31 }
32
33
34
```

114. w.a.p strong number?



```
1 package Practice;
2
3 import java.util.Scanner;
4
5 public class StrongNumber
6 {
7     static int fact(int n)
8     {
9         int fact=1;
10        while (n>0)
11        {
12            fact= fact*n;
13            n--;
14        }
15        return fact;
16    }
17    public static void main(String[] args)
18    {
19        Scanner sc=new Scanner(System.in);
20        System.out.println("enter a number");
21        int n=sc.nextInt();
22        int num=n;
23        int sum=0;
24        int t=num;
25        while(num!=0)
26        {
27            int r=num%10;
28            sum=sum+fact(r);
29            num=num/10;
30        }
31
32        if (sum==t)
33            System.out.println(t + " is a strong number");
34        else
35
36            System.out.println(t + " not a strong number");
37    }
38 }
```

Problems @ Javadoc Declaration Console

<terminated> StrongNumber [Java Application] C:\Program Files\Java\jdk1.8.0_162\bin\javaw.exe (22-Sep-2019)

enter a number

145

145 is a strong number