```
package in.pwskills;
public class Sample
{
      static final String name = new String("sachin");
}

Write a code to find the length of the String present in name variable
      Sample.name.length()


package java.lang;

import java.io.PrintWriter;
class System
{
      public static final Writer out = new PrintWriter();
}

package java.io;
class PrintWriter
{
      println(){

      }
      println(int){

      }
      println(String){

      }
      print(int){

      }
      print(){

      }
}

UserCode
++++++++
import java.lang.*; //default package avaialable to Compiler and JVM

public class Test{
      public static void main(String[] args){
                  System.out.println("Welcome to java coding....");
      }
}


Serialization and DeSerialization
++++++++++++++++++++++++++++++++
Agenda :
1.  Serialization
2.  Deserialization
3.  transient keyword
4.  static Vs transient
5.  transient Vs final
6.  Object graph in serialization.
7.  customized serialization.
```

8.  Serialization with respect inheritance.
9.  Externalization
10. Difference between Serialization & Externalization
11. SerialVersionUID


Serialization: (1.1 v)
=>  The process of saving (or) writing state of an object to a file is called serialization but strictly speaking it is the process of converting an
     object from java supported  form to either network supported form (or) file supported form.

=>  By using FileOutputStream and ObjectOutputStream classes we can achieve serialization process.
                              |=> writeObject(Object obj)
       Ex: Myntra product


De-Serialization:
=>  The process of reading state of an object from a file is called DeSerialization but strictly speaking it is the process of converting an
       object from file supported form (or) network supported form to java supported form.

=>  By using FileInputStream and ObjectInputStream classes we can achieve DeSerialization.
                              |=> readObject()


eg#1.
```
import java.io.*;

//NotSerializableException
class Dog  implements Serializable
{
      int i =10;
      int j =20;
}

public class Test
{
   public static void main(String[] args) throws Exception
      {
            Dog d1 = new Dog();

            System.out.println("****Serialization Started****");

            FileOutputStream fos   = new FileOutputStream("Dog.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            System.out.println("Dog Object data :: "+d1.i+"-----"+d1.j);
            oos.writeObject(d1);

            System.out.println("****Serialziation Completed****");

            System.out.println();

            System.out.println("****De-Serialization Started****");

            FileInputStream fis   = new FileInputStream("Dog.ser");
```

```
            ObjectInputStream ois = new ObjectInputStream(fis);
            Dog d2 = (Dog)ois.readObject();
            System.out.println("Dog Object data :: "+d2.i+"-----"+d2.j);

            System.out.println("****De-Serialization Completed****");

      }
}

Output
****Serialization Started****
Dog Object data :: 10-----20
****Serialziation Completed****

****De-Serialization Started****
Dog Object data :: 10-----20
****De-Serialization Completed****

eg#2.
import java.io.*;

//NotSerializableException
class Dog  implements Serializable
{
      int i =10;
      int j =20;
}
class Cat implements Serializable{
      int i = 100;
      int j = 200;
}

public class Test
{
   public static void main(String[] args) throws Exception
      {
            Dog d1 = new Dog();
            Cat c1 = new Cat();

            System.out.println("****Serialization Started****");

            FileOutputStream fos   = new FileOutputStream("obj.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            System.out.println("Dog Object data :: "+d1.i+"-----"+d1.j);
            System.out.println("Cat Object data :: "+c1.i+"-----"+c1.j);
            oos.writeObject(d1);
            oos.writeObject(c1);

            System.out.println("****Serialziation Completed****");

            System.out.println();

            System.out.println("****De-Serialization Started****");

            FileInputStream fis   = new FileInputStream("obj.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Cat c2 = (Cat)ois.readObject();
            Dog d2 = (Dog)ois.readObject();
            System.out.println("Dog Object data :: "+d2.i+"-----"+d2.j);
```

```
            System.out.println("Cat Object data :: "+c2.i+"-----"+c2.j);

            System.out.println("****De-Serialization Completed****");

      }
}
```

Output

```
****Serialization Started****
Dog Object data :: 10-----20
Cat Object data :: 100-----200
****Serialziation Completed****

****De-Serialization Started****
Exception in thread "main" java.lang.ClassCastException: Dog cannot be cast to Cat
```

Note:

1. We can perform Serialization only for Serilizable objects.
2. An object is said to be Serilizable if and only if the corresponding class implements Serializable interface.
3. Serializable interface present in java.io package and does not contain any methods. It is marker interface.
   The required ability will be provided automatically by JVM.
4. We can add any no. Of objects to the file and we can read all those objects from the file but in which order we wrote objects in the
   same order only the objects will come back. That is order is important.
5. If we are trying to serialize a non-serializable object then we will get RuntimeException saying "NotSerializableException".

Transient keyword:

1. transient is the modifier applicable only for variables,but not for classes and methods.
2. While performing serialization if we don't want to save the value of a particular variable to meet security constant such
   type of variable ,then we should declare that variable with "transient" keyword.
3. At the time of serialization JVM ignores the original value of transient variable and save default value to the file .
4. That is transient means "not to serialize"

static Vs transient :

1. static variable is not part of object state hence they won't participate in serialization because of this declaring a static variable as
    transient there is no use.

Transient Vs Final:

1. final variables will be participated into serialization directly by their values.
     Hence declaring a final variable as transient there is no use.
       //the compiler assign the value to final variable

```
import java.io.*;

//NotSerializableException
class Dog  implements Serializable
{
       transient final int i =10;//class variables
       transient static int j =20;//value will particpate not the variable
```

```
}

public class Test
{
   public static void main(String[] args) throws Exception
      {
            Dog d1 = new Dog();

            System.out.println("****Serialization Started****");

            FileOutputStream fos   = new FileOutputStream("obj.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            System.out.println("Dog Object data :: "+d1.i+"-----"+d1.j);
            oos.writeObject(d1);

            System.out.println("****Serialziation Completed****");

            System.out.println();

            System.out.println("****De-Serialization Started****");

            FileInputStream fis   = new FileInputStream("obj.ser");
            ObjectInputStream ois = new ObjectInputStream(fis);
            Dog d2 = (Dog)ois.readObject();
            System.out.println("Dog Object data :: "+d2.i+"-----"+d2.j);

            System.out.println("****De-Serialization Completed****");

      }
}
Output
declaration output
int i=10;
int j=20;
10.................20

transient int i=10;
int j=20;
0.................20

transient int i=10;
transient static int j=20;
0.................20

transient final int i=10;
transient int j=20;
10.................0

transient final int i=10;
transient static int j=20;
10.................20



Tuesday         : 7.30PM to 10.30PM (topic: GarbageCollector)
Wedenesday      : 7.30PM to 10.30PM (topic: Serialization and DeSerialization)
Friday          : No Session
Saturday,Sunday : Normal timings
```