INTERN REPORT - FEB-19

Programming basics - Classpath

1. What is a process and what is a thread?

- Process means any program is in execution.
- Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc.
- Process takes more time to terminate and it is isolated means it does not share memory with any other process.
- The process can have the following states like new, ready, running, waiting, terminated, suspended.

- ❏ Thread is the segment of a process.
- ❏ A process can have multiple threads and these multiple threads are contained within a process.
- ❏ A thread has 3 states: running, ready, and blocked.
- ❏ Thread takes less time to terminate as compared to process.

2. Open Task Manager and observe the running processes and its attributes like process ID, memory used, CPU used, disk usage, network usage and number of threads. And understand what these numbers mean.

- ➔ CPU - the mathematical ability of the computer
- ➔ MEMORY - RAM, temporary, fast
- ➔ DISK - the speed of reading and writing data from hard drive
- ➔ NETWORK - computers ability to communicate over network based on network card(the port where data is sent in and out to internet)
- ➔ GPU - graphics card usage

3. What are Environmental Variables? Significance of PATH variables

- ➢ envi variable is those whose value is set outside the program i.e operating system
- ➢ Each variable has NAME=value
- ➢ NAME will be uppercase, value will be string
- ➢ any number of envi variables can be created and used at any point of time.
- ➢ primary use is to limit the need of modifying and re-releasing apps due to changes in config data.

- ➢ PATH specifies set of dir where executable codes are located
- ➢ PATH contains string of dir separated by semicolon

4. Difference between 32 bit and 64 bit applications

- ➔ 32bit system - 32 bit address representation - 2^32 memory addresses - 4GB
- ➔ 64bit system - 64 bit address representation - 2^64 memory addresses - 16 EXAbytes

5. Write a small java program (say a "Hello World" program) and put it in a directory. Compile the program and change directory to some other directory and try to execute the class file (This will help you to learn about setting PATH/CLASSPATH variables!).

```
public class q5
{
        public static void main(String a[])
        {
                System.out.println("Hello world in javapr");
        }
}
```

OUTPUT:

```
C:\Users\Yasin\javapr\s1_q5>javac q5.java
C:\Users\Yasin\javapr\s1_q5>set CLASSPATH=C:\Users\Yasin\javapr\s1_q5\temp
C:\Users\Yasin\javapr\s1_q5>java q5
Hello world in javapr
```

6. Write a java program to print the size of different data types in java (list all the data types!).

```
public class q6
{
  public static void main (String[]args)
  {
        System.out.println ("Size of byte is " + Byte.SIZE / 8 + " Byte");
        System.out.println ("Size of char is " + Character.SIZE / 8 + " Bytes");
        System.out.println ("Size of short is " + Short.SIZE / 8 + " Bytes");
        System.out.println ("Size of integer is " + Integer.SIZE / 8 + " Bytes");
        System.out.println ("Size of long is " + Long.SIZE / 8 + " Bytes");
        System.out.println ("Size of float is " + Float.SIZE / 8 + " Bytes");
        System.out.println ("Size of double is " + Double.SIZE / 8 + " Bytes");
  }
}
```

OUTPUT:

```
Size of byte is 1 Byte
Size of char is 2 Bytes
Size of short is 2 Bytes
Size of integer is 4 Bytes
Size of long is 8 Bytes
Size of float is 4 Bytes
Size of double is 8 Bytes
```

7. Write a program to read Input data from command line.

```
public class q7
{
   public static void main(String[] args) {
        for(int i=0;i<args.length;i++)
        {
        System.out.println(args[i]);
        }
   }
}
```

COMMAND LINE ARGUMENTS: yasin ss zohocorp com

OUTPUT:

```
yasin
ss
zohocorp
com
```

8. Write a class with a package statement in it and try to run it.

Syntax for compiling a package : javac -d directory filename.java

```
package samp;
public class PackSamp
{
        public static void main(String a[])
        {
                System.out.println("Hello world in javapr");
        }
}
```

OUTPUT:

```
C:\Users\Yasin\javapr>javac -d . PackSamp.java

C:\Users\Yasin\javapr>java PackSamp
Error: Could not find or load main class PackSamp
Caused by: java.lang.NoClassDefFoundError: samp/PackSamp (wrong name:
PackSamp)

C:\Users\Yasin\javapr>java samp.PackSamp
Hello world in javapr
```

9. Write two classes (say "A" and "B") , in class "A" create an instance of class "B". Compile class "A" in some directory "D1" and class "B" in "D2" , now A.class and B.class are in two separate directories. Try to run class "A" from directory "D1" (it will say class "B" not found). If you set classpath to both directories then it will work (try it out!)

D1/A.java:

```
public class A
{
	public static void main(String a[])
	{
		System.out.println("Hello world in A");
		B bobj=new B();
		bobj.disp();
	}
}
```

D2/B.java:

```
public class B
{
	public void disp()
	{
		System.out.println("Hello world in B");
	}
}
```

OUTPUT:

```
O:\Zoho\intern docs\day1\javapr\s1_q9>cd D2

O:\Zoho\intern docs\day1\javapr\s1_q9\D2>javac B.java

O:\Zoho\intern docs\day1\javapr\s1_q9\D2>cd ..

O:\Zoho\intern docs\day1\javapr\s1_q9>cd D1

O:\Zoho\intern docs\day1\javapr\s1_q9\D1>javac -cp ../D2/ A.java

O:\Zoho\intern docs\day1\javapr\s1_q9\D1>cd ..

O:\Zoho\intern docs\day1\javapr\s1_q9>java -cp D1;D2; A
Hello world in A
Hello world in B
```

10. Set the classpath from the class "A" itself (instead of specifying the classpath in the DOS shell , try to set in in your java program!)
Solution folder : s1_q10

11. Try to learn the difference between String and StringBuffer classes by writing small programs.

- ➔ String objects are immutable, StringBuffer objects are mutable.
- ➔ String is slow and consumes more memory, StringBuffer fast and consumes less memory.
- ➔ If the content is fixed and won't change frequently, we should go with string,
- ➔ If content keeps on changing , if thread safety is required, then go with stringbuffer.

- ➔ If content keeps on changing , and if thread safety is not required, then go with stringbuilder.

NOTE: When multiple threads are working on the same data, and the value of our data is changing, that scenario is not thread-safe.

```
class StvsBuf
{
	public static void main(String a[])
	{
		String s1="Hello";
		StringBuffer s2= new StringBuffer("Hi");

		s1+=" world"; // new memory space is created
		s2.append(" world"); //updated on the same memory location

		System.out.println(s1+" "+s2);
	}
}
```

OUTPUT:

```
C:\Users\Yasin\javapr\s1_11>javac StvsBuf.java

C:\Users\Yasin\javapr\s1_11>java StvsBuf
Hello world Hi world
```

12. Write a java program to reverse a string using StringBuffer class (do not use String class for data manipulation!)

```
class Main
{
	public static void main(String []a)
	{
	StringBuffer ys= new StringBuffer("zoho corporation");
	System.out.println("Original string :"+ys);
	ys.reverse();
	System.out.println("Reversed string :"+ys);
	}
}
```

OUTPUT:

```
Original string :zoho corporation
Reversed string :noitaroproc ohoz
```

Access Specifiers

1. Learning about access specifiers (Write java programs to bring about the difference between , public ,private,package and protected specifiers!)
Solution folder :s2_q1

	Public	protected	package	private
Within class	yes	yes	yes	yes
Within package child class	yes	yes	yes	no
Within package Non Child class	yes	yes	yes	no
Outside package Child class	yes	yes	no	no
Outside package Non Child class	yes	no	no	no

2. What is the meaning of declaring a class public? (try this by putting the class in a package and try to access the class from a different package!).
Solution folder : s2_q2

Target.java:

```
package classSpecifier;
public class Target
{
	public void Disp()
	{
		System.out.println("Inside target class");
	}
}
```

From.java:

```
package classSpecifier2;
import classSpecifier.*;
public class From
```

```
{
	public static void main(String a[])
	{
		//System.out.println("Inside from class");
		Target tc= new Target();
		tc.Disp();
	}
}
```

OUTPUT:

```
C:\Users\Yasin\javapr\PublicClass>javac -d . Target.java

C:\Users\Yasin\javapr\PublicClass>javac -d . From.java

C:\Users\Yasin\javapr\PublicClass>java classSpecifier2.From
Inside target class
```

3. Learning about scope specifiers also what is the difference between instance variables and class variables? Write programs and try using local scope/method scope/member scope variables and how to access them.

<u>**Instance variables:**</u>

- ❏ Instance variables are declared in a class but outside a method or any block.
- ❏ Instance variables are created only when an object of the class is created and deleted when the object is deleted.
- ❏ Instance variables can be accessed by using
  syntax: *objectRefernce.variableName;*

<u>**Class variables:**</u>

- ❏ Class variable or static variables are variables which are declared with a keyword static in a class but outside a method or any block.
- ❏ Class variables are created once the program is executed and deleted when execution of the program comes to an end.
- ❏ Class variables can be accessed by
  Syntax : *className.variableName;*

4. Write a java program to print the system properties (Hint : Refer to System class) Try to print the current working directory from your java program.

- For every system, some info will be maintained in the form of properties.
- Properties class is a subclass of hashtable.
- getProperties() to get all the properties from properties file.
- getProperty(key) to get a particular property.
- Main advantage of system property is we can customize the program behavior without recompiling.

```
import java.util.Properties;
class SysProp
{
	public static void main(String a[])
	{
		//Properties prop = System.getProperties();
		//prop.list(System.out);

		String dir=System.getProperty("user.dir");
		System.out.print(dir);
	}
}
```

OUTPUT:

```
C:\Users\Yasin\javapr
```

5. Is the method main belong to any class? , if not why? If so, what are the access privileges to that method?

- ➔ Compiler won't be responsible for checking the main method, JVM always searches for the main method at runtime.
  - ◆ Public - to be called by JVM from anywhere.
  - ◆ Static - JVM has to call this method without object creation.
  - ◆ Void - main method won't return anything to JVM.
  - ◆ main() - main() is the word that is configured in JVM to be executed first.
  - ◆ String[] args - optional command line arguments.
- ➔ Main method does not belong to any particular class.Also since it's static it does not belong to any particular object.
- ➔ There is only one access privilege for the main method [public] else JVM does not consider it's an entry point. JVM looks for the *public static void main* with an optional argument to start the code.

6. Is it possible to have a private class? (write a class and try it out! , try protected also!)

- ➔ Top level class can't be private or protected.
- ➔ Inner class can be.

```
C:\Users\Yasin\javapr>javac PriClass.java                    //private class
PriClass.java:2: error: modifier private not allowed here
private class PriClass
        ^
1 error

C:\Users\Yasin\javapr>javac PriClass.java                    //protected class
PriClass.java:2: error: modifier protected not allowed here
protected class PriClass
          ^
1 error
```

```
C:\Users\Yasin\javapr>javac PriClass.java                    //default

C:\Users\Yasin\javapr>javac PriClass.java                    //public class
```

7. Is it possible to have a private constructor? (write a class and try it out!) , if possible, how do you create an instance of it from another class?

- ➔ Private constructor is used for creating a singleton class.
- ➔ Singleton class is allowed to have only one object.

```
class test
{
        static test obj = new test();
        private test()
        {

        }
        public static test getIns()
        {
                return obj;
        }
}


public class q7
{
        public static void main(String a[])
        {
                test obj1 = test.getIns();
        }
}
```

8. What is the meaning of declaring a variable "static" ? (write a program to bring out the difference between instance and class variables).

- ➔ Static variable in Java is a variable which belongs to the class and initialized only once at the start of the execution.
- ➔ It is a variable which belongs to the class and not to object(instance).
- ➔ Static variables are initialized only once, at the start of the execution.
- ➔ A single copy to be shared by all instances of the class.
- ➔ A static variable can be accessed directly by the class name and doesn't need any object.
    - ◆ className.staticVariable

```
class One
{
        int insVar=1;
        static int classVar=2;
}
class Two
{
        public static void main(String a[])
        {
                System.out.println(One.classVar);          //static or class variable

                One x=new One();
                System.out.println(x.insVar);              //instance variable
        }
}
```

OUTPUT:

```
C:\Users\Yasin\javapr>java Two
2
1
```

9. What is the meaning of declaring a method "static" ? how do you invoke such a method ?

- ➔ Static method in Java is a method which belongs to the class and not to the object.
- ➔ A static method can access only static data. It cannot access non-static data (instance variables).
- ➔ A static method can call only other static methods and can not call a non-static method from it.
- ➔ A static method can be accessed directly by the class name and doesn’t need any object

```
class One
{
        int insVar=1;
        static int classVar=2;

        static void prClaVar()
        {
                System.out.println(classVar);
        }
}

class StMethod
{
        public static void main(String a[])
        {
                One.prClaVar();
        }
}
```

OUTPUT:

```
C:\Users\Yasin\javapr>java StMethod
2
```

10. What is the meaning of declaring a class static?

- ➔ Static modifier is not allowed in top level class, inner classes can be static.
- ➔ A static inner class is a nested class which is a static member of the outer class.
- ➔ It can be accessed without instantiating the outer class, using other static members.
- ➔ Just like static members, a static nested class does not have access to the instance variables and methods of the outer class.

11. What is the meaning of declaring a class final?

- ➔ A final class is simply a class that **can't be extended**.
- ➔ It does not mean that all references to objects of the class would act as if they were declared as final.
- ➔ Good reasons to prohibit inheriting the class.

12. How do you prevent overriding of a method (can you prevent overloading of a method?)

We can prevent method overriding by following ways,

- ➔ Using a static method
- ➔ Using private access modifier
- ➔ Using default access modifier
- ➔ Using the final keyword method

Method overloading cannot be prevented in java.Basically it's like creating a new method.

13. What are inner classes and anonymous classes? Why are they used?

<u>Inner class:</u>

- ➔ Java inner class or nested class is a class which is declared inside the class or interface.
- ➔ We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.
- ➔ Additionally, it can access all the members of the outer class including private data members and methods.

<u>Anonymous class:</u>

- ➔ Anonymous inner class is a type of inner class which has no name and can have only one object.
- ➔ This class is mainly used to override methods of its class or interface without subclass.

14. Is there any difference between an inner class and nested class?

<u>Inner class</u>

Non static class
outerClass obj = new outerClass();
outerClass.nestedClass obj1 = obj.new nestedClass();

<u>Nested class</u>

Static class
outerClass.nestedClass obj = new outerClass.nestedClass();

Advanced Concepts

1. Convert a float value to an integer (float to double etc.) (Hint use wrapper classes, do not typecast them!).

- ➔ Wrapper class provides a mechanism to convert primitive to object(Autoboxing) as well as object to primitive(Unboxing).

```
class q1
{
	public static void main(String a[])
	{
		Float fobj=4.5f;
		Integer iobj=fobj.intValue();
		System.out.println(iobj);

		Double dobj = fobj.doubleValue();
		System.out.println(dobj);
	}
}
```

2. Function overriding & overloading (Write a set of classes to bring out the difference!)

Overriding:
- ➔ If the method of the parent class is redefined by child class, then this is called overriding.

Overloading:
- ➔ Two methods are said to be overloaded if both have the same name but different argument types or different number of arguments.

```
class Parent {
   void show()
   {
      System.out.println("Parent's function");
   }
}

class Child extends Parent {
   void show()
   {
      System.out.println("Child's function");
   }
   void disp(int x)
   {
        System.out.println("normal disp function");
   }
```

```
    void disp(int x,int y)
    {
         System.out.println("overloaded function");
    }
}

class OrOl {
    public static void main(String[] args)
    {

        Parent obj1 = new Parent();
        obj1.show();

        Child obj2 = new Child();
        obj2.show();

        obj2.disp(1);
        obj2.disp(1,1);
    }
}
```
OUTPUT:
```
C:\Users\Yasin\javapr>java OrOl
Parent's function
Child's function
normal disp function
overloaded function
```

3. Write a class and override "toString" method in the class object. (When you write a class "A" does it have any parent? , investigate!).

Every class in java has a default parent class ‘Object’.

```
class ComplexNum {
    private int re, im;
    public ComplexNum(int re, int im) {
        this.re = re;
        this.im = im;
    }

    public String toString() {
        return re + "+" + im+"i";
    }
}
public class q3 {
    public static void main(String[] args) {
        ComplexNum c1 = new ComplexNum(5,4);
        System.out.println(c1);
    }
}
```

4. What is an abstract class , why is it used ? (Write an abstract and try to use it!).

- ➔ If the implementation of a class is not complete, then such a **partially implemented class** can be declared as an abstract class.
- ➔ If a class contains at least 1 abstract object , then the class must be declared as abstract class.
- ➔ Abstract class can contain 0 number of abstract methods also,no one is able to create objects or access the methods of an abstract class. We can declare the class as abstract to prevent creation of objects.

.

```
abstract class base{
        abstract void disp();
}
class test1 extends base{
        void disp(){
                System.out.println("Overridden by test1 class");
        }
}
class test2 extends base{
        void disp(){
                System.out.println("Overridden by test2 class");
        }
}
class q4{
        public static void main(String[] args) {

                //base x = new base();        //base is abstract; cannot be instantiated
                //x.disp();
                base a= new test1();
                a.disp();
                base b = new test2();
                b.disp();
        }
}
```

5. Write an interface and try to bring out diff. between abstract class and an interface! (can I define variables in interfaces?)

- ➔ Abstract class can have both abstract and non-abstract class. But interfaces can have only abstract methods.
- ➔ Abstract class does not support multiple inheritance, but interface does.
- ➔ Abstract classes can have any variables, but interfaces can have only static or final variables.

```
interface intf
{
        static String s = "static var of interface";   //That's should static or final
        void disp();
}
class imp implements intf{
	public void disp()
	{
		System.out.println(s);
	}
}
class q5{
	public static void main(String[] args)
	{
		intf obj = new imp();
		obj.disp();
	}
}
```

6. What is the difference between a Hashtable and Properties? (Write a program as an example!)

<u>Hash Table</u>

- A Hashtable is a key-value pair.
- Java Hashtable class contains unique elements.
- Java Hashtable class doesn't allow null key or value.

<u>Properties</u>

- The Properties class provides methods to get data from the properties file and store data into the properties file.
- it can be used to get the properties of a system.
- Recompilation is not required if the information is changed from a properties file.

7. Write a program to print all elements in a Properties object (Using Enumeration!)

```
import java.util.Properties;
import java.util.Enumeration;

class PropEnum
{
	public static void main(String a[])
	{
		Properties ps = System.getProperties();
		Enumeration es = ps.propertyNames();

		while(es.hasMoreElements())
		{
			String sc= (String) es.nextElement();
			System.out.println(sc + " : " + ps.getProperty(sc));
		}
	}
}
```

OUTPUT:

```
C:\Users\Yasin\javapr>javac PropEnum.java

C:\Users\Yasin\javapr>java PropEnum
java.runtime.name : Java(TM) SE Runtime Environment
java.vm.version : 15.0.2+7-27
sun.boot.library.path : C:\Program Files\Java\jdk-15.0.2\bin
java.vm.vendor : Oracle Corporation
java.vendor.url : https://java.oracle.com/
path.separator : ;
java.vm.name : Java HotSpot(TM) 64-Bit Server VM
sun.os.patch.level :
user.script :
..
.// many other properties
```

8. Can you remove an element while enumerating through a Properties object? (If so how? If not how will I remove an element while enumerating!)

Enumeration does not have a method to remove an object.

For removing elements while traversing, iterator or listIterator can be used instead of enum.

9. How do you catch an exception? Write a program to catch a NullPointerException!

An exception can be caught by using a catch clause with a defined set of exceptions to be caught and their respective actions.

```
class NulEx
{
	public static void main(String a[])
	{
		String st=null;
		try
		{
			if(st.equals("yasin"))
			{
				System.out.println("same");
			}
		}
		catch(NullPointerException ne)
		{
			System.out.println("Null pointer exception caught here!!");
		}
	}
}
```

10. How do you throw an exception?

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exceptions.

The throw keyword is mainly used to throw custom exceptions.

throw new exceptionName("User exception");

11. What is the difference between an Error and an Exception?

<u>EXCEPTION</u>

- ➔ Exceptions are caused mostly by our programs.
- ➔ We can recover from exception by either try-catch block or throwing exceptions back to the user.
- ➔ Ex : ArrayIndexOutOfBoundException

<u>ERROR</u>

- ➔ Indicates serious problems that a reasonable application should not try to catch.
- ➔ Errors are mostly caused by lack of system resources.
- ➔ Recovering from error is not possible.
- ➔ Ex: StackOverflowError

12. In java are the parameters passed by reference or by value (Write a program to test for basic data types , arrays ,strings ,string arrays and custom objects !)

- ➔ In java, primitive datatypes and strings are passed by value.
- ➔ Arrays and other objects are passed by reference.

```
class Data
{
        int a;
        String b;
        Data()
        {
                a=1;
                b="Hello world";
        }
}
class PassBy
{
        static void chInt(int i)
        {
                i++;
        }
        static void chFl(float i)
        {
                i++;
        }
        static void chStArr(String arr[])
        {
                arr[1]="changed";
        }
        static void chObj(Data d)
        {
                d.a=0;
                d.b="changed";
        }
        public static void main(String a[])
        {
                int intvar=1;
                float flvar=1.0f;
                System.out.println("Actual values before passing :"+intvar+", "+flvar);
                chInt(intvar);
                chFl(flvar);
                System.out.println("Values After passing :"+intvar+",  "+flvar+"\n");
                //therefore primitive datatypes and string are passed as value

                String[] arr= {"hi","hello","vanakam","welcome","namasthae"};
                System.out.print("Actual values before passing :");
                for(int i=0;i<5;i++)
                {
                        System.out.print(arr[i]+" ");
                }
                System.out.println();
                chStArr(arr);
                System.out.print("Values After passing :");
                for(int i=0;i<5;i++)
                {
                        System.out.print(arr[i]+" ");
                }
                System.out.println("\n");
                Data obj1=new Data();
                System.out.println("Actual values before passing :"+obj1.a+" "+obj1.b);
                chObj(obj1);
                System.out.println("Values after passing :"+obj1.a+" "+obj1.b);
                //therefore objects are passed by reference
        }
}
```

OUTPUT:

```
C:\Users\Yasin\javapr>java PassBy
Actual values before passing :1, 1.0
Values After passing :1, 1.0

Actual values before passing :hi hello vanakam welcome namasthae
Values After passing :hi changed vanakam welcome namasthae

Actual values before passing :1 Hello world
Values after passing :0 changed
```

<u>I/O Operations</u>

1. Bring out differences between char streams and byte streams.

<u>CHAR STREAMS:</u>

- ➔ Character stream automatically allows us to read/write data character by character(16 bits).
- ➔ For example FileReader and FileWriter are character streams used to read from source and write to destination.
- ➔ Character stream is useful when we want to process text files.

<u>BYTE STREAMS:</u>

- ➔ Byte streams process data byte by byte (8 bits).
- ➔ For example FileInputStream is used to read from source and FileOutputStream to write to the destination.
- ➔ A byte stream is suitable for processing raw data like binary files..

2. What is the format of char data type and why it is platform independent?

- ➔ It stores character constants.
- ➔ It is of a size of 2 bytes(16bits), but basically it can hold only a single character because char stores unicode character sets.
- ➔ It has a minimum value of ‘u0000’ (or 0) and a maximum value of ‘uffff’ (or 65,535, inclusive).
- ➔ Char steam is of UTF-16 format and thereby it covers all the recognized characters so it is platform independent.

3. How do you convert char streams to byte streams?

We can convert char stream to byte stream by using OutputStreamWriter class. And by the same way InputStreamWriter is used to convert byte stream to char stream.

```
import java.io.*;
public class chtoby
{
   public static void main(String a[]) throws IOException
   {
      FileReader in = new FileReader( "from.txt" );
      FileOutputStream out = new FileOutputStream( "to.txt" );
      int c;
      while((c=in.read())!=-1)         //reads as char stream
      {
        out.write(c);                  //writes as byte stream
      }
      in.close();
      out.close();
   }
}
```

4. Write a program to read from a lengthy string (using streams) and write the output to a file.

```
import java.io.*;
class filewr
{
	public static void main(String a[]) throws IOException
	{
		FileReader in = new FileReader("from.txt");
		FileWriter out = new FileWriter("to.txt");
		int c;
		while((c= in.read())!=-1)
		{
			out.write(c);
		}
		System.out.println("Successfully written");
		in.close();
		out.close();
	}
}
```

5. Write a program to redirect your print statements (System.out and System.err) to two files stdout.txt and stderr.txt.

```
import java.io.*;
class q5
{
	public static void main(String a[]) throws IOException
	{
		PrintStream op = new PrintStream("stdout.txt");
		PrintStream er = new PrintStream("stderr.txt");
		System.setOut(op);
		System.setErr(er);
		System.out.println("written on stdout.txt file");
		System.err.println("written on stderr.txt file");
	}
}
```

6. Write a program to read a text file (using byte streams) convert the bytes to chars and print the output.

```
import java.io.*;
class q6
{
	public static void main(String a[]) throws IOException
	{
		FileInputStream rd = new FileInputStream("from.txt");
		int bn;
		while((bn= rd.read())!=-1)
		{
			//System.out.print(bn);			// prints  byte stream
			System.out.print((char)bn);		// prints char stream
		}
		rd.close();
	}
}
```

7.Write a program to get the result for the following
In a school(1-10),each class is maintaining a file having data about all the students in a format (Roll###Name###Age###phone).Principal want to generate a file containing data of all the students who is above 10 years old in a format(Name##Age##phone##class)

```
import java.io.*;
class q7
{
	public static void main(String a[]) throws IOException
	{
		FileWriter out  = new FileWriter("output.txt");
		for(int i=1;i<=10;i++)
		{
			FileReader in = new FileReader("class"+i+".txt");
			BufferedReader bf = new BufferedReader(in);
			String s1;
			while((s1 = bf.readLine())!= null)
			{
				String[] res = s1.split("###");
				int x = Integer.parseInt(res[2]);
				if(x>10)
				{
					String temp = 
res[1]+"##"+res[2]+"##"+res[3]+"##class"+i+"\n";
					out.write(temp);
				}
			}in.close();
		}
		out.close();
	}
}
```