EXPERIMENT NO 10

1. What is use of Console class?

Ans: The Java Console class is be used to get input from console. It provides methods to read texts and passwords. If you read password using Console class, it will not be displayed to the user. The java.io.Console class is attached with system console internally.

Simple example to read text from console.

```
String text=System.console().readLine();
System.out.println("Text is: "+text);
```

Let's see the declaration for java.io.Console class:

```
public final class Console extends Object implements Flushable
```

System class provides a static method console() that returns the singleton instance of Console class.

2. What is the use of System class?

Ans: The java.lang.System class contains several useful class fields and methods. It cannot be instantiated. Facilities provided by System -

- Standard output
- Error output streams
- Standard input and access to externally defined properties and envi- ronment variables.
- A utility method for quickly copying a portion of an array.
- A means of loading files and libraries.

Following is the declaration for java.lang.System class -

3. How to create file using append mode in java?

```
public final class System
   extends Object
```

Ans: In Java we can append a string in an existing file using FileWriter which has an option to open file in append mode. Java FileWriter class is used to write character-oriented data to a file. It is character-oriented class which is used for file handling in Java. Unlike FileOutputStream class, we don't need to convert string into byte array because it provides method to write string directly.

Note: The buffer size may be specified, or the default size may be used. A Writer sends its output immediately to the underlying character or byte stream.

Class constructor used:

FileWriter(File file, boolean append) constructs a FileWriter object given a File object in append mode.

Class methods used:

void write(String s, int off, int len)

This method writes a portion of a String.

DKTE

```java
// Java program to append a string to the
// end of a file.
import java.io.*;

public class GeeksforGeeks {

    public static void appendStrToFile(String fileName,
                                       String str)
    {
        try {

            // Open given file in append mode.
            BufferedWriter out = new BufferedWriter(
                    new FileWriter(fileName, true));
            out.write(str);
            out.close();
        }
        catch (IOException e) {
            System.out.println("exception occoured" + e);
        }
    }

    public static void main(String[] args)
        throws Exception
    {
        // Let us create a sample file with some text
        String fileName = "Geek.txt";
        try {
            BufferedWriter out = new BufferedWriter(
                        new FileWriter(fileName));
            out.write("Hello World:\n");
            out.close();
        }
        catch (IOException e) {
            System.out.println("Exception Occurred" + e);
        }

        // Let us append given str to above
        // created file.
        String str = "This is GeeksforGeeks";
        appendStrToFile(fileName, str);

        // Let us print modified file
        try {
            BufferedReader in = new BufferedReader(
                        new FileReader("Geek.txt"));

            String mystring;
            while ((mystring = in.readLine()) != null) {
                System.out.println(mystring);
            }
        }
        catch (IOException e) {
            System.out.println("Exception Occurred" + e);
        }
    }
}
```

4. How to increase the buffer size?

Ans: Java BufferedWriter class

Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits Writer class. The buffering characters are used for providing the efficient writing of single arrays,characters, and strings.

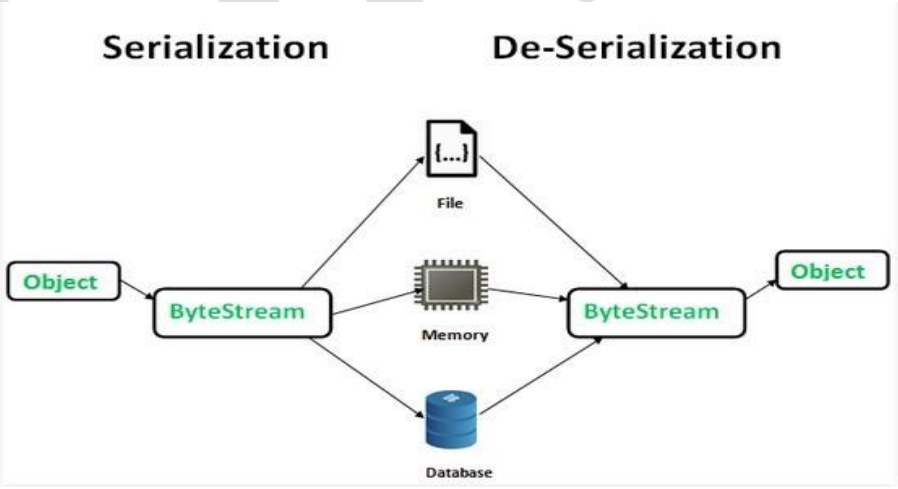
Class constructors -

Constructor	Description
BufferedWriter(Writer wrt)	It is used to create a buffered character output stream that uses the default size for an output buffer.
BufferedWriter(Writer wrt, int size)	It is used to create a buffered character output stream that uses the specified size for an output buffer.

We can use second constructor to specify the size of buffer. Similar constructors are also available for BufferedReader class, BufferedInputStream class and BufferedOutputStream class.

5. What is serialization in Java? Explain in detail with example.

Ans: Serialization in Java is a mechanism of writing the state of an object into a byte stream. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies. The reverse operation of serialization is called deserialization.



The byte stream created is platform independent. So, the object serialized on one platform can be deserialized on a different platform.

java.io.Serializable interface

To make a Java object serializable we implement the java.io.Serializable interface.

Serializable is a marker interface (has no data member and method). It is used to ”mark” Java classes so that objects of these classes may get the certain capability. The Cloneable and Remote are also marker interfaces.

It must be implemented by the class whose object you want to persist.

The String class and all the wrapper classes implement the java.io.Serializable interface by default.

ObjectOutputStream class:

The ObjectOutputStream class is used to write primitive data types, and Java objects to an OutputStream. Only objects that support the java.io.Serializable interface can be written to streams.

Constructor -

1) public ObjectOutputStream(OutputStream out) throws IOException {}	creates an ObjectOutputStream that writes to the specified OutputStream.

Important Methods -

Method	Description
1) public final void writeObject(Object obj) throws IOException {}	writes the specified object to the ObjectOutputStream.
2) public void flush() throws IOException {}	flushes the current output stream.
3) public void close() throws IOException {}	closes the current output stream.

Example of Java Serialization -

```
import java.io.Serializable;
public class Student implements Serializable{
 int id;
 String name;
 public Student(int id, String name) {
  this.id = id;
  this.name = name;
 }
}
```

```
import java.io.*;
class Persist{
 public static void main(String args[])throws Exception{
  Student s1 =new Student(211,"ravi");

  FileOutputStream fout=new FileOutputStream("f.txt");
  ObjectOutputStream out=new ObjectOutputStream(fout);

  out.writeObject(s1);
  out.flush();
  System.out.println("success");
 }
}
```

Points to remember -

1. If a parent class has implemented Serializable interface then child class doesn't need to implement it but vice-versa is not true.
2. Only non-static data members are saved via Serialization process.
3. Static data members and transient data members are not saved via Serialization process. So, if you don't want to save value of a nonstatic data member then make it transient.
4. Constructor of object is never called when an object is deserialized.
5. Associated objects must be implementing Serializable interface.