**Java\_excer3**

Java I/O Problems –

1. Write Java program to read input from java console.

import java.io.Console;

class ReadStringTest{

public static void main(String args[]){

Console c=System.console();

System.out.println("Enter your College Name: ");

String n=c.readLine();

System.out.println("Welcome "+n);

}

}

1. Write a java program to read a file line by line and store it into a variable.

import java.io.\*;

public class ReadLineByLineExample1

{

public static void main(String args[])

{

try

{

File file=new File("Demo.txt"); //creates a new file instance

FileReader fr=new FileReader(file); //reads the file

BufferedReader br=new BufferedReader(fr); //creates a buffering character input stream

StringBuffer sb=new StringBuffer(); //constructs a string buffer with no characters

String line;

while((line=br.readLine())!=null)

{

sb.append(line); //appends line to string buffer

sb.append("\n"); //line feed

}

fr.close(); //closes the stream and release the resources

System.out.println("Contents of File: ");

System.out.println(sb.toString()); //returns a string that textually represents the object

}

catch(IOException e)

{

e.printStackTrace();

}

}

}

1. How would you append data to the end of a file? Show the constructor for the class you would use.

Append text to end of file using echo command: echo 'text here' >> filename.

1. Suppose you wanted to write code that reads from a file one word at a time. The code needs to peek ahead to find where the words are separated by whitespace. What input stream could you use to accomplish this and how ?

You can use the java.util.StringTokenizer (in the API reference documentation) or java.io.StreamTokenizer (in the API reference documentation) to parse your input into words. Each class has a default set of delimiters (like white space) that you can specify.

1. Implement a pair of classes, one Reader and one Writer, that count the number of times a particular character, such as e, is read or written. The character can be specified when the stream is created. Write a program to test your classes.
2. Write a program named carddWriter that creates a card instance, displays its value, and serializes it into a file named card.out. Here is an example of what CardWriter might display:

Multithreading

1. write a Java program that uses a pair of threads to simulate withdrawal/deposit of financial transactions. In that program, one thread performs deposits while the other performs withdrawals. Each thread manipulates a pair of shared variables, class and instance field variables, that identifies the financial transaction's name and amount. For a correct financial transaction, each thread must finish assigning values to the name and amount variables (and print those values, to simulate saving the transaction) before the other thread starts assigning values to name and amount (and also printing those values).

ANS] // NeedForSynchronizationDemo.java

class NeedForSynchronizationDemo

{

public static void main (String [] args)

{

FinTrans ft = new FinTrans ();

TransThread tt1 = new TransThread (ft, "Deposit Thread");

TransThread tt2 = new TransThread (ft, "Withdrawal Thread");

tt1.start ();

tt2.start ();

}

}

class FinTrans

{

public static String transName;

public static double amount;

}

class TransThread extends Thread

{

private FinTrans ft;

TransThread (FinTrans ft, String name)

{

super (name); // Save thread's name

this.ft = ft; // Save reference to financial transaction object

}

public void run ()

{

for (int i = 0; i < 100; i++)

{

if (getName ().equals ("Deposit Thread"))

{

// Start of deposit thread's critical code section

ft.transName = "Deposit";

try

{

Thread.sleep ((int) (Math.random () \* 1000));

}

catch (InterruptedException e)

{

}

ft.amount = 2000.0;

System.out.println (ft.transName + " " + ft.amount);

// End of deposit thread's critical code section

}

else

{

// Start of withdrawal thread's critical code section

ft.transName = "Withdrawal";

try

{

Thread.sleep ((int) (Math.random () \* 1000));

}

catch (InterruptedException e)

{

}

ft.amount = 250.0;

System.out.println (ft.transName + " " + ft.amount);

// End of withdrawal thread's critical code section

}

}

}

}