Java technologies essay sample

Q. 1: a)What is Object Oriented Paradigm? Explain advantages of Object Oriented Programming. 5 marks

Ans Object Oriented paradigm: The Object Oriented paradigm is centered on the concept of the object. Everything is focused on objects. In this language, program consists of two things: first, a set of objects and second the way they interact with each other. Computation in this paradigm is viewed as the simulation of real world entities. The popular programming languages in this paradigm are C++, Simula, Smalltalk and Java

Object Oriented programming: The world is Object Oriented, and Object Oriented programming expresses programs in the ways that model how people perceive the world. Figure 2 shows different real world objects around us which we often use for performing different functions. This shows that problem solving using the objects oriented approach is very close to our real life problem solving techniques. The basic difference in Object Oriented programming (OOP) is that the program is organized around the data being operated upon rather than the operations performed. The basic idea behind OOP is to combine both, data and its functions that operate on the data into a single unit called object. Object Oriented methods are favored because many experts agree that Object Oriented techniques are more disciplined than conventional structured techniques. (Martin and Odell 1992)

b) What is polymorphism? Explain the advantages of polymorphismwith an example. Solution : Polymorphism is the capability of a method to do different things based on the object through which it is invoked or object it is acting upon. For example method find _area will work definitely for Circle object and Triangle object In Java, the type of actual object always determines method calls; object reference type doesn't play any role in it. You have already used two types of polymorphism (overloading and overriding) in the previous unit and in the current unit of this block. Now we will look at the third: dynamic method binding. Java uses Dynamic Method Dispatch mechanism to decide at run time which overridden function will be invoked. Dynamic Method Dispatch mechanism is important because it is used to implement runtime polymorphism in Java. Java uses the principle: " a super class object can refer to a subclass object" to resolve calls to overridden methods at run time. If a superclass has method that is overridden by its subclasses, then the different versions of the overridden methods are invoked or executed with the help of a superclass reference variable. Assume that three subclasses (Cricket_Player Hockey_Player and Football_Player) that derive from Player abstract class are defined with each subclass having its own Play() method. abstract class Player // class is abstract

```
{
private String name;
public Player(String nm)
{
name= nm;
}
public String getName() // regular method
{
return (name);
}
```

```
public abstract void Play();

// abstract method: no implementation

}

class Cricket_Player extends Player

{

Cricket_Player( String var)

{

}

public void Play()

{

System. out. println(“ Play Cricket:”+getName());

}

}

class Hockey_Player extends Player

{

Hockey_Player( String var)

{

}

public void Play()

{

System. out. println(“ Play Hockey:”+getName());

}

}

class Football_Player extends Player

{

Football_Player( String var)
```

```
{
}
public void Play()
{
System. out. println(“ Play Football:”+getName());
}
}
public class PolyDemo
{
public static void main(String[] args)
{
Player ref; // set up var for an Playerl
Cricket_Player aCplayer = new Cricket_Player(“ Sachin”); // makes specific
objects Hockey_Player aHplayer = new Hockey_Player(“ Dhanaraj”);
Football_Player aFplayer = new Football_Player(“ Bhutia”);
// now reference each as an Animal
ref = aCplayer;
ref. Play();
ref = aHplayer;
ref. Play();
ref = aFplayer;
ref. Play();
}
}
```

```
Output: Play Cricket: Sachin
Play Hockey: Dhanaraj
```

Play Football: Bhutia

Question 2: a) What is platform independence? Explain why java is secure and

platform independent. (3 Marks)

b) Write a program in java to generate Fibonnaci Series. (3 Marks) c) Explain the advantage of of Unicode. (2 Marks)

d) Explain the significance of PATH and CLASS PATH. (2 Marks) Solution (a): PlatformIndependent

Java is Platform independent. The meaning of platform here may be confusing for you but actually this word is poorly defined. In the computer industry it typically means some combination of hardware and system software but here you can understand it as your operating system. Java is compiled to an intermediate form called Java byte-code or simply byte code. A Java program never really executes immediately after compilation on the host machine. Rather, this special program called the Java interpreter or Java VirtualMachine reads the byte code, translates it into the corresponding host machine instructions and then executes the machine instruction.

A Java programcan run on any computer systemfor which a JVM (Java Virtual Machine) and some library routines have been installed. The second important part which makes Java portable is the elimination of hardware architecture dependent constructs. For example, Integers are always four bytes long and floating-point variables followthe IEEE 754. You don’t need to worry that the interpretation of your integer is going to change if you move from one hardware to another hardware like Pentiumto a PowerPC. We can develop the Java program on any computer system and the execution of that program is possible on any other computer system loaded with JVM. For example, we can write and compile the Java program on Windows 98 and execute the compiled program on JVM of the Macintosh operating system.

Solution (b): Solution

```
Input – 8
Output – 1 1 2 3 5 8 13 21 */
class Fibonacci{
public static void main(String a rgs[]){
int num = Integer. parseInt(args[0]); //taking no. as command line argument.

System. out. println("*****Fibonacci Series*****");
int f1, f2= 0, f3= 1;
for (int i= 1; i <= num; i++){
System. out. print(" "+f3+" ");
f1 = f2;
f2 = f3;
f3 = f1 + f2;
}
}
```

Solution (c): Unicode is a 16-bit code having a large range in comparison to previous ASCII code, which is only 7 bits or 8 bits in size. Unicode can represent all the characters of all human languages. Since Java is developed for designing Internet applications, and worldwide people can write programs in Java, transformation of one language to another is simple and efficient. Use of Unicode also supports platformindependence in Java.

Solution (d): PATH variable

In JDK the PATH variable contains directories where binary files (e. g. EXE files in Windows) will be looked for. We set the PATH variables like this i. e path C: Javajdk1. 6. 0_03in (i)on command prompt

C:> set path=%path; C: Javajdk1. 6. 0_03in%

When you open a command prompt and type “ javac”, you’re supposed to have the “ bin” directory of your sdk into the PATH, otherwise you’ll get an infamous “ Command not found” error message CLASSPATH

In JDK the CLASSPATH contains directories (or JAR files), from where your java compiler/runtime will look for . class files (and some others). For example, “ java Hello. class” will not work unless you set the directory (or JAR file) Hello. class is in, into your CLASSPATH. i. e. classpath C: Javajdk1. 6. 0_03lib

For setting CLASSPATH using command prompt

Java class path can be set using either the -classpath option when calling an SDK tool (the preferred method) or by setting the CLASSPATH environment variable. The -classpath option is preferred because you can set it individually for each application without affecting other applications and without other applications modifying its value. (ii)on command prompt

C:> set classpath=%classpath; C: Javajdk1. 6. 0_03lib%

Question 3: a) What is an exception? Create an exception subclass named MyException to handle user generate exceptions in Java. (5 Marks) b) What is abstract class? What are advantages of using abstract class? Write a program in Java to explain abstract class and multilevel inheritance. (5 Marks)

Solution (a): Java uses exceptions as a way of signaling serious problems when you execute a program. The standard classes use them extensively. Since they arise in your Java programs when things go wrong, and if something can go wrong in your code, sooner or later it will, they are a very basic consideration when you are designing and writing your programs.

An exception usually signals an error and is so called because errors in your Java programs are bound to be the exception rather than the rule—by definition! An exception doesn't always indicate an error though—it can also signal some particularly unusual event in your program that deserves special attention.

An exception in Java is an object that's created when an abnormal situation arises in your program. This exception object has fields that store information about the nature of the problem. The exception is said to be thrown—that is, the object identifying the exceptional circumstance is tossed as an argument to a specific piece of program code that has been written specifically to deal with that kind of problem. The code receiving the exception object as a parameter is said to catch it. An example of how to define an exception class:

```
public class DreadfulProblemException extends Exception
{
// Constructors
public DreadfulProblemException(){ } // Default constructor
public DreadfulProblemException(String s)
{
super(s); // Call the base class constructor
}
}
```

Solution (b): An abstract class is a class in which one or more methods are declared, but not defined. The bodies of these methods are omitted, because, as in the case of the method sound() in the Animal class, implementing the methods does not make sense. Since they have no definition and cannot be executed, they are called abstract methods. The declaration for an abstract method ends with a semicolon and you specify the method with the keyword abstract to identify it as such. To declare that a class is abstract you just use the keyword abstract in front of the class keyword in the first line of the class definition.

A program in Java to explain abstract class and multilevel inheritance

```
abstract class Player // class is abstract
{
private String name;
public Player(String nm)
{
name= nm;
}
public String getName() // regular method
{
return (name);
}
public abstract void Play();
// abstract method: no implementation
}
class Cricket_Player extends Player
{
Cricket_Player( String var)
{}
public void Play()
{
System. out. println(“ Play Cricket:”+getName());
}
}
class Hockey_Player extends Player
{
Hockey_Player( String var)
{
}
public void Play()
{
System. out. println(“ Play Hockey:”+getName());
}
}
class Football_Player extends Player
{
Football_Player( String var)
```

```
{
}
public void Play()
{
System. out. println(“ Play Football:”+getName());
}
}
public class PolyDemo
{
public static void main(String[] args)
{
Player ref; // set up var for an Playerl
Cricket_Player aCplayer = new Cricket_Player(“ Sachin”); // makes specific
objects Hockey_Player aHplayer = new Hockey_Player(“ Dhanaraj”);
Football_Player aFplayer = new Football_Player(“ Bhutia”);
// now reference each as an Animal
ref = aCplayer;
ref. Play();
ref = aHplayer;
ref. Play();
ref = aFplayer;
ref. Play();
}
}
```

Output:

```
Play Cricket: Sachin
```

Play Hockey: Dhanaraj

Play Football: Bhutia

Question 4: Differentiate the following and support with example: (10 Marks)

Final and static member

Inheritance and Aggregation

Abstract class and Interface

String and String Buffer

Solution: (I) Final and staticmember

Solution

1. Static variables (also called class variables) only exist in the class they are defined in. They are not instantiated when an instance of the class is created. In other words, the values of these variables are not a part of the state of any object. When the class is loaded, static variables are initialized to their default values if no explicit initialization expression is specified Final variable: values of final variables cannot be changed.

2. Static methods are also known as class methods. A static method in a class can directly access other static members in the class. It cannot access instance (i. e., non-static) members of the class, as there is no notion of an object associated with a static method. However, note that a static method in a class can always use a reference of the class's type to access its members, regardless of whether these members are static or not.

3. Final methods: cannot be overriden.

FinalMember

static final variables are not the same with final(non-static) variables!

Final(non-static) variables can differ from object to object!!! But that's only if the initialization is made within the constructor! (If it is not initialized from the constructor then it is only a waste of memory as it creates final variables for every object that is created that cannot be altered.) (ii) Inheritance and Aggregation

There are two schools of thought on how to best extend, enhance, and reuse code in an objectoriented system:
1. Inheritance: extend the functionality of a class by creating a subclass. Override superclass members in the subclasses to provide new functionality. Make methods abstract/virtual to force subclasses to " fill-in-the-blanks" when the superclass wants a particular interface but is agnostic about its implementation.

2. Aggregation: create new functionality by taking other classes and combining them into a new class. Attach an common interface to this new class for interoperability with other code.
It's not a matter of which is the best, but of when to use what. (iii) Abstract class and Interface
An interface is an empty shell, there are only the signatures (name / params / return type) of the methods. The methods do not contain anything. The interface can't do anything. It's just a pattern Implementing an interface consume very little CPU, because it's not a class, just a bunch of names, and therefor there is no expensive lookup to do. It's great when it matters such as in embedded devices.

Abstract classes
Abstract classes, unlike interfaces, are classes. There are more expensive to

use because there is a lookup to do when you inherit fromthem.

Abstract classes look a lot like interfaces, but they have something more : you can define a behavior for them. It’s more about a guy saying “ these classes should look like that, and they got that in common, so fill in the blanks !”.

(iv) String and String Buffer

A String is immutable, i. e. when it’s created, it can never change. A StringBuffer (or its non-synchronized cousin StringBuilder) is used when you need to construct a string piece by piece without the performance overhead of constructing lots of little Strings along the way.

Question 5: a) What are the classes in Java available for file handling? Write a program in Java to append content at the end of an
already existing file. (5 Marks)
b)Explain the difference between checked and unchecked
exceptions with example.
Solution a) Solution:
Java input and output is based on the use of streams, or sequences of bytes that travel from a source to a destination over a communication path. If a program is writing to a stream, you can consider it as stream’ s source. If it is reading froma stream, it is the stream’ s destination. The communication path is dependent on the type of I/O being performed. It can consist of memory-to-memory transfers, a file system, a network, and other forms of I/O. File handling in java is available through streams and stream classes

The Java model for I/O is entirely based on streams. There are two types of streams: byte streams and character streams.

• Byt e str eams carry integers with values that range from 0 to 255. A diversified data can be expressed in byte format, including numerical data, executable programs, and byte codes – the class file that runs a Java program. • Char acter Str eams are specialized type of byte streams that can handle only textual data. Most of the functionality available for byte streams is also provided for character streams. The methods for character streams generally accept parameters of data type char, while byte streams work with byte data types. The names of the methods in both sets of classes are almost identical except for the suffix, that is, character-stream classes end with the suffix Reader or Writer and byte-stream classes end with the suffix InputStream and OutputStream. For example, to read files using character streams use the Java. io. FileReader class, and for reading it using byte streams use Java. io. FileInputStream

A Programto Copy an already existing File to new File

```
import java. io.*;
public class jCOPY {
public static void main(String args[]){
try{
jCOPY j = new jCOPY();
j. CopyFile(new File(args[0]), new File(args[1]));
}
catch (Exception e) {
e. printStackTrace();
}
}
public void CopyFile(File in, File out) throws
Exception { FileInputStream fis = new
FileInputStream(in); FileOutputStream fos = new
FileOutputStream(out);
byte[] buf = new byte[1024];
int i = 0;
while((i= fis. read(buf))!=-
1) { fos. write(buf, 0, i);
}
fis. close();
fos. close();
}
}
```

ii) Solution:

As stated by their name, unchecked exceptions are not checked at compile-time which means that the compiler doesn't require methods to catch or to specify (with a throws) them. Classes belonging to this category are detailed in the section 11. 2 Compile-Time Checking of Exceptions of the JLS:

The unchecked exceptions classes are the class RuntimeException and its subclasses, and the class Error and its subclasses. All other exception classes are checked exception classes. The Java API defines a number of exception classes, both checked and unchecked. Additional exception classes, both checked and unchecked, may be declared by programmers. See §11. 5 for a description of the exception class hierarchy and some of the exceptionclasses defined by the Java API and Java virtual machine

Question 6: a) What is multithreading? Explain the two ways of creating threads in Java programs. Also explain difference between
notify() and notify All( ) methods. (5 Marks)
b) What is need of Layout Manager? Explain different layouts available in Java. (5 Marks)
Solution a):
Multithreaded programs support more than one concurrent thread of execution. This means they are able to simultaneously execute multiple sequences of instructions. Each instruction sequence has its own unique flow of control that is independent of all others. These independently executed instruction sequences are known as threads.

Your PC has only a single CPU; you mi ght ask how it can execute more than one thread at the same time? In single processor systems, only a single thread of execution occurs at a given instant. But multiple threads in a programincrease the utilization of CPU.

The CPU quickly switches back and forth between several threads to create an illusion that the threads are executing at the same time. You know that single-processor systems support logical concurrency only. Physical concurrency is not supported by it. Logical concurrency is the characteristic exhibited when multiple threads execute with separate, independent flow of control. On the other hand on a multiprocessor system, several threads can execute at the same time, and physical concurrency is achieved.

Creating Threads in Java

The multithreading system in Java is built upon the Thr ead Class, its methods and its companion interface, Runnable. To create a new thread, your program will either extend Thread Class or implement the Runnable interface. The Thread Class defines several methods that help in managing threads. For example, if you have to create your own thread then you have to do one of the following 1)

```
classMyThread extends Thread
{
MyThread(arguments) // constructor
{
} //initialization
public void run()
{
// performoperations
}
}
```

Write the following code to create a thread and start it running: MyThread p = newMyThread(arguments);

```
p. start();
```

2)

```
classMyThread implements Runnable
{
MyThread(arguments)
{
```

```
//initialization
}
public void run()
{
// performoperation
}
}
```

notify(): this method should be called only when the current thread has already acquired the lock on the object. If the wait set of the object is non-empty then a thread from the set is arbitrarily chosen, removed and is re-enabled for thread scheduling. Of course, the thread will not be able to proceed unless the current thread releases the object’s lock.

notifyAll(): it’s same as the notify() method. The only difference is that in this case all the threads from the non-empty wait set of the object are removed and are re-enabled for thread scheduling in stead of only one thread from the wait set being picked arbitrarily, removed, and re-enabled for thread scheduling as is the case in notify() method.

b) A LayoutManager rearranges the components in the container based on their size relative to the size of the container.
Consider the window that just popped up. It has got five buttons of varying sizes. Resize the window and watch how the buttons move. In particular try making it just wide enough so that all the buttons fit on one line. Then try making it narrow and tall so that there is only one button on line. See if you can manage to cover up some of the buttons. Then uncover them. Note that

whatever you try to do, the order of the buttons is maintained in a logical way. When you add a component to an applet or a container, the container uses its layout manager to decide where to put the component. Different LayoutManager classes use different rules to place components. java. awt. LayoutManager is an interface. Five classes in the java packages implement it:

• FlowLayout • BorderLayout • CardLayout •

GridLayout

• GridBagLayout

•plus javax. swing. BoxL ayout

Question 7: a) What is an Applet? Write an applet that prints “ Lear Java it is useful” at the current cursor position whenever the mouse left button is clicked. (5 Marks)

b) Consider a class that stores a Bank account holder’s name, account number, ATM card number, account balance and

ATM PIN. Write a program to store the data onto a disk file, except for the account balance and ATM PIN. Use serialization and transient variables. (5 Marks) Solution a):

Applet is a Java program that runs in the Appletviewer (a test utility for Applets that is included with the J2SDK) or a World Wide Web browser such as Microsoft Internet Explorer or Netscape Communicator.

The Applet class is packed in the Java. Applet package which has several interfaces. These interfaces enable the creation of Applets, interaction of Applets with the browser, and playing audio clips in Applets. In Java 2, class

Javax. swing. JApplet is used to define an Applet that uses the Swing GUI components.

// An Applet program having two text boxes and one button. Read your name in one text box // and when button is pressed then your name is transferred into text box two import java. awt.*;

```
import java. applet.*;
import java. awt. event.*;
public class Q6a extends Applet implements ActionListener{
TextField text1, output;
Label label1, label2;
Button button;
public void init(){
setLayout(null);
label1 = new Label(“ Enter Name: “);
label1. setBounds(20, 20, 100, 20);
add(label1);
text1 = newTextField(5);
text1. setBounds(150, 20, 100, 20);
add(text1);
label2
label2 = new Label(“ You Entered: “);
label2. setBounds(20, 80, 130, 20);
add(label2);
output = new TextField(5);
output. setBounds(150, 80, 100, 20);
add(output);

button = new Button(“ Submit”);

button. setBounds(150, 110, 100, 20);

add(button);

button. addActionListener(this);

}

public void actionPerformed(ActionEvent ae){

String src= text1. getText();

output. setText(src);

}
```

Question 8: a) Write a Java program to set up JDBC and execute the following SQL statement on a database table employee-t with the fields emp-id, emp name, emp-department, empbasic

“ SELECT * FROM employee-t where emp-basic < 10000. (7 Marks) b) What is Java Bean ? What are its advantages? (3 Marks)

Solution:

Java Database Connectivity (JDBC) is a class library which provides a standard way for establishing and maintaining a Java program’ s connection to a database.

Java provides JDBC to connect to databases and work with it. Using standard library routines, you can open a connection to the database. Basically JDBC allows the integration of SQL calls into a general programming environment by providing library routines, which interface with the database. In particular, Java’ s JDBC has a rich collection of routines which makes such an interface extremely simple and intuitive.

```
Class. forName(“ sun. jdbc. odbc. JdbcOdbcDriver”)
Connection con = DriverManager. getConnection(url, username, password);
Statement stmt = con. createStatement();
ResultSet rs = stmt. executeQuery(“ SELECT * FROM employee-t where emp-
basic < 10000”); while ( rs. next() )
{
ename = rs. getString(“ emp_name”);
eaddress = rs. getString(“ emp_address”);
esal = rs. getFloat(“ emp_salary”);
System. out. println(ename + ” address is” + eaddress + ” draws salary ” +
esal + “ in dollars”); }
```

b) A JavaBean on its own is not terribly interesting, it’s just a Java class that conforms to some standards that you listed above. However, conformance with this standard is one of the pillars on which the Java EE framework is built and it comes up in quite a few places. I suspect that when you hear about all of the great things that JavaBeans can do, what’s being referred to in Enterprise JavaBeans (EJBs).