SOLVED PAST PAPER INTERNET PROGRAMMING 2015

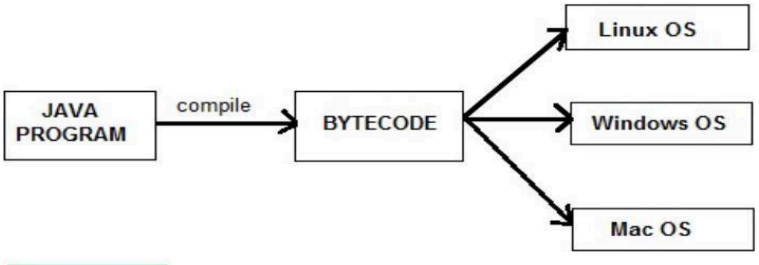
Q2. Short Answers:

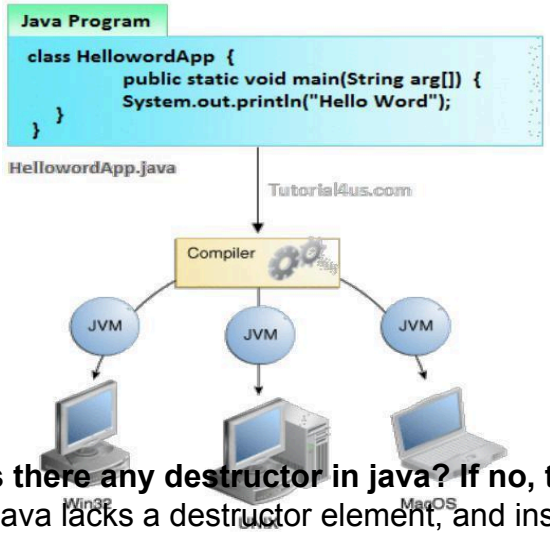
What do you mean by "Java is platform independent"?

A: We say Java is a platform-independent language in the sense that a program written in Java can be executed on any operating system with any hardware. Now, of course, many programs we are all familiar with, such as Skype or Excel, can be used on multiple operating systems like Windows 8 and OS X. But notice that these programs have different versions for different operating systems, which means that they are actually not platform independent. For programs written in Java, you can directly use them on either Windows or Mac OS or even Linux without any modification. This is a special feature of Java.

It means the same Java program can be run on any platform or operating system e.g. Windows, Linux or Solaris without any change.

- Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere (WORA).





2. Is there any destructor in java? If no, then how will the object be destroyed?

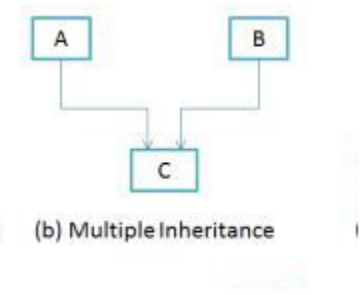
A: Java lacks a destructor element, and instead uses a garbage collector for resource deallocation. It is possible to use something similar to a destructor in Java, the method Object.finalize(), which however does not work exactly as a standard destructor would.

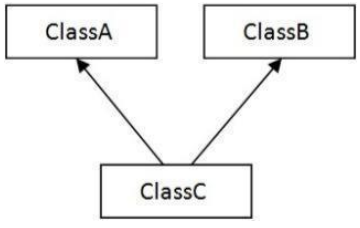
There is no destructor in java.

- ✔ Java performs garbage collection and eliminates the need to free objects explicitly. When an object has no references to it anywhere except in other objects that are also unreferenced, its space can be reclaimed.
- ✔ Before an object is destroyed, it might be necessary for the object to perform some action. For example: to close an opened file. In such a case, define a *finalize()* method with the actions to be performed before the object is destroyed.

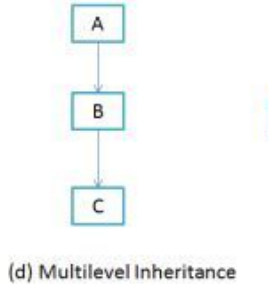
3. What is meant by multiple inheritance and multi-level inheritance?

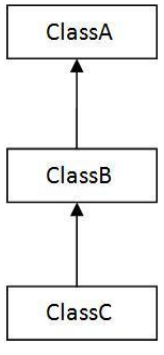
A: “Multiple Inheritance” refers to the concept of one class extending (Or inherits) more than one base class. The inheritance we learnt earlier had the concept of one base class or parent. The problem with “multiple inheritance” is that the derived class will have to manage the dependency on two base classes.





Multilevel inheritance refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A. For more details and example refer – Multilevel inheritance in Java.





4. Describe the two attributes i.e. action and method of form tag in html?

A:
```
<form action="demo_form.asp" method="get">
 First name: <input type="text" name="fname"><br>
 Last name: <input type="text" name="lname"><br>
 <input type="submit" value="Submit">
</form>
```

Action Attribute

The action attribute specifies where to send the form-data when a form is submitted.

Syntax

<form action="*URL*">

Attribute Values

Value	Description
URL	Where to send the form-data when the form is submitted. Possible values: • An absolute URL - points to another web site (like action="http://www.example.com/example.htm") • A relative URL - points to a file within a web site (like action="example.htm")

Method attribute

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).

The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").

Syntax

<form method="get|post">

Attribute Values

Value	Description
get	Default. Appends the form-data to the URL in name/value pairs: URL?name=value&name=value
post	Sends the form-data as an HTTP post transaction

5. How does java perform event handling?
A: Steps involved in event handling:
The User clicks the button and the event is generated.
Now the object of concerned event class is created automatically and information about the source and the event get populated with in same object.
Event object is forwarded to the method of registered listener class.
The method is now get executed and returns.
There are many ways to handle events. One of them is…

By Implementing Listener Interfaces
Java defines interfaces for every event type
If a class needs to handle an event. It needs to implement the corresponding listener interface
To handle “ActionEvent” a class needs to implement “ActionListener”
To handle “KeyEvent” a class needs to implement “KeyListener”
To handle “MouseEvent” a class needs to implement “MouseListener” and so on

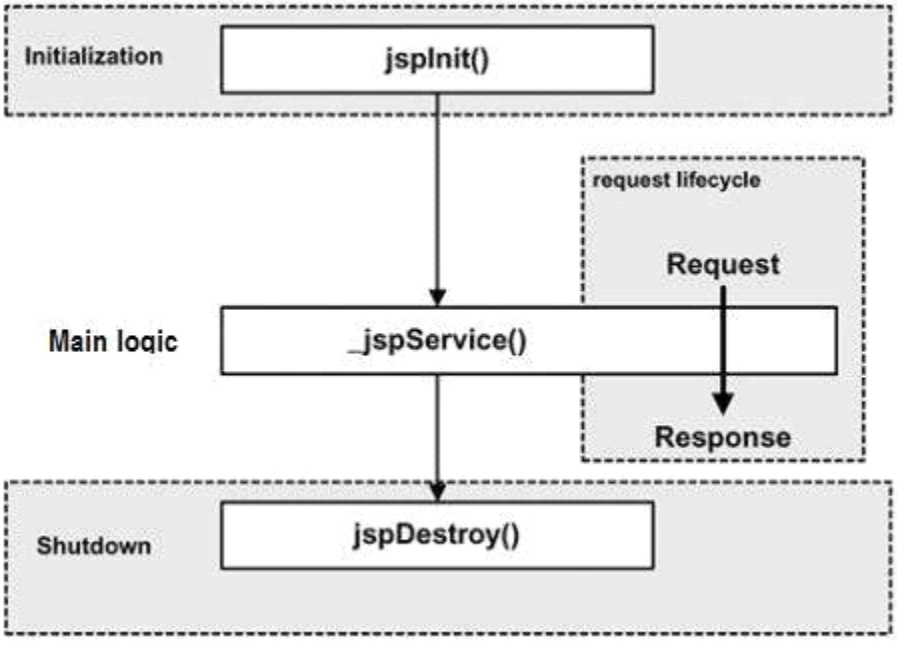
Event Handling Steps

Step 1: Create components which can generate events (Event Generators)
Step 2: Build component (objects) that can handle events (Event Handlers)
Step 3: Register handlers with generators

6. Writes down a life cycle of JSP page?
A: A JSP life cycle can be defined as the entire process from its creation till the destruction which is similar to a servlet life cycle with an additional step which is required to compile a JSP into servlet.
The following are the paths followed by a JSP

- ✔ Compilation
- ✔ Initialization
- ✔ Execution

Cleanup
The four major phases of JSP life cycle are very similar to Servlet Life Cycle and they are as follows:



7. What is difference between package and public access modifiers?

Package

A **java package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

- ✔ Built-in Package:-Existing Java package for example java.lang, java.util etc.
- ✔ User-defined-package:- Java package created by user to categorized classes and interface

Public access modifier

The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.

Default access modifier***(confused whether it is package or not)

If you don't use any modifier, it is treated as **default** bydefault. The default modifier is accessible only within package.

Example of default access modifier

In this example, we have created two packages pack and mypack. We are accessing the A class from outside its package, since A class is not public, so it cannot be accessed from outside the package.

```
//save by A.java
package pack;
class A{
 void msg(){System.out.println("Hello");}
}
```

Public access modifier

The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.

Example of public access modifier

```
//save by A.java

package pack;
public class A{
public void msg(){System.out.println("Hello");}
}
```

Output:Hello

8. Why PHP, ASP.NET, JSP etc. are known as Server-side scripting language?

A: PHP, ASP.NET,JSP etc. are known as Server-side scripting language

- ✔ Server side scripting means that all of the code is executed on the server before the data is passed to the user's browser.(http://php.about.com/od/programingglossary/g/server_side.htm)
- ✔ **Server-side scripting** is a technique used in web development which involves employing scripts on a web server which produce a response customized for each user's (client's) request to the website. The alternative is for the web server itself to deliver a static web page.(https://en.wikipedia.org/wiki/Server-side_scripting#Languages)

OR

Server-side scripting is a technique used in web development which involves employing **scripts** on a web **server** which produce a response customized for each user's (client's) request to the website. JSP, ASP.NET and PHP are server-side because that is where the logic of their content generation is executed. So when a PHP, ASP.NET or JSP page is requested, the server executes the code there and sends the results, an HTML page usually, back to the requesting client. The client simply displays the results.

9. What is meant by a scripting language?

A: A scripting language is a programming language designed for integrating and communicating with other programming languages. Some of the most widely used scripting languages are JavaScript, VBScript, PHP, Perl,

Python, Ruby, ASP and Tcl. Since a scripting language is normally used in conjunction with another programming language, they are often found alongside HTML, Java or C++…

A scripting language is a programming language that employs a high-level construct to interpret and execute one command at a time. In general, scripting languages are easier to learn and faster to code in than more structured and compiled languages such as C and C++.

10. Describe ResultSet object used in JDBC?

A: A ResultSet object maintains a cursor pointing to its current row of data. Initially the cursor is positioned before the first row. The next method moves the cursor to the next row, and because it returns false when there are no more rows in the ResultSet object, it can be used in a while loop to iterate through the result set.

A ResultSet object maintains a cursor that points to the current row in the result set. The term "result set" refers to the row and column data contained in a ResultSet object.

The methods of the ResultSet interface can be broken down into three categories −

Navigational methods: Used to move the cursor around.

Get methods: Used to view the data in the columns of the current row being pointed by the cursor.

Update methods: Used to update the data in the columns of the current row. The updates can then be updated in the underlying database as well.

JDBC provides the following connection methods to create statements with desired ResultSet −

createStatement(int RSType, int RSConcurrency);

prepareStatement(String SQL, int RSType, int RSConcurrency);

prepareCall(String sql, int RSType, int RSConcurrency);

Question # 3

Briefly write down the answers of the following questions.

1. **What is meant by interface and abstract classes? Explain with an example program.**

   An **interface in java** is a blueprint of a class. It has static constants and abstract methods only.

   The interface in java is **a mechanism to achieve fully abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

   Java Interface also **represents IS-A relationship**

   It cannot be instantiated just like abstract

Simple example of Java interface

In this example, Printable interface have only one method, its implementation is provided in the A class.

```
1. interface printable{
2. void print();
3. }
4.
5. class A6 implements printable{
6. public void print(){System.out.println("Hello");}
7.
8. public static void main(String args[]){
9. A6 obj = new A6();
```

```
10. obj.print();
11.  }
12. }
```

=>Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only important things to the user and hides the internal details for example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated.

Example of abstract class that has abstract method

In this example, Bike the abstract class that contains only one abstract method run. It implementation is provided by the Honda class.

```
1.  abstract class Bike{
2.    abstract void run();
3.  }
4.  class Honda4 extends Bike{
5.  void run(){System.out.println("running safely..");}
6.  public static void main(String args[]){
7.   Bike obj = new Honda4();
8.   obj.run();
9.  }
10. }
```

OR

ABSTRACT CLASS

Definition:

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods (method with body).

Ways to achieve Abstaction

There are two ways to achieve abstraction in java

1. Abstract class (0 to 100%)
2. Interface (100%)

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

A class that is declared as abstract is known as **abstract class**. It needs to be extended and its method implemented. It cannot be instantiated.

Example abstract class

abstract class **A{}**

Abstract method

A method that is declared as abstract and does not have implementation is known as abstract method.

Example abstract method

```
abstract void printStatus();//no body and abstract
```

Interface

Definition:

An **interface in java** is a blueprint of a class. It has static constants and abstract methods only.

The interface in java is **a mechanism to achieve fully abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

Implementing (using) Interface

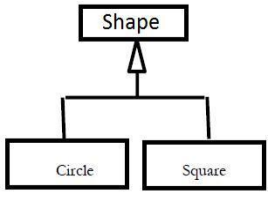
Classes *implement* interfaces.

Implementing an interface is like *signing a contract*.

A class that implements an interface will have to provide the definition of all the methods that are present inside an interface

If the class does not provide definitions of all methods, the class would not compile. We have to declare it as an abstract class in order to get it compiled.

EXAMPLE PROGRAM FOR ABSTRACT CLASS



The **Shape** class contains an abstract method calculateArea() with no definition.

```
public abstract class Shape{
public abstract void calculateArea();
}
```

Class **Circle** extends from abstract Shape class, therefore to become concrete class it must provides the definition of calculateArea() method.

```
public class Circle extends Shape {
private int x, y;
private int radius; public
Circle() {
x = 5;
y = 5;
radius = 10;
}
```

```
// providing definition of abstract method
public void calculateArea () {
double area = 3.14 * (radius * radius);
System.out.println("Area: " + area);
}
}//end of class
```

The **Test class** contains main method. Inside main, a reference s of abstract Shape class is created. This reference can point to Circle (subclass of abstract class Shape) class object as it is a concrete class. With the help of reference, method calculateArea() can be invoked of Circle class. This is all shown in the form of code below

```
public class Test {
public static void main(String args[]){ //can
only create references of A.C.
Shape s = null;
//Shape s1 = new Shape(); //cannot instantiate //abstract
class reference can point to concrete subclass
s = new Circle();
s.calculateArea();
}
}//end of class
```

EXAMPLE PROGRAM FOR INTERFACE

```
public interface Speaker{
public void speak();

}
```

Interface Speaker is implemented by three classes Politician, Coach and Lecturer. Code snippets of all these three classes are show below:

```
public class Politician implements Speaker{
public void speak(){
System.out.println("Politics Talks");
}
}
public class Coach implements Speaker{
public void speak(){
System.out.println("Sports Talks");
}
}
public class Lecturer implements Speaker{
public void speak(){
System.out.println("Web Design and Development Talks"); }
}

public class Test{
public static void main (String args[ ]) {
Speaker sp = null;
System.out.println("sp pointing to Politician");
sp = new Politician();
sp.speak();
System.out.println("sp pointing to Coach");
sp = new Coach();
sp.speak();
System.out.println("sp pointing to Lecturer");
sp = new Lecturer();
sp.speak();
}
```

}

2. A client wants to send a request to server for checking that a string is palindrome or not. So, you have to make a server-client application, in which client sends a string to the server and server resends the response in which it tells that the given string is palindrome or not. (Do your task with serialization method in which either you can use built-in class String or make your own string class).

Client.java import java.net.*; import java.io.*; import javax.swing.*; public class Client{
public static void main(String args[]){ try{
Socket s=new Socket("localhost",2222);
OutputStream os=s.getOutputStream();
ObjectOutputStream oos=new ObjectOutputStream(os);
String inputString;
inputString=JOptionPane.showInputDialog("Enter a string");
Writing object to network oos.writeObject(inputString);
InputStream is=s.getInputStream();
ObjectInputStream ois=new ObjectInputStream(is);
// Read String Object from network
String stringFromServer=(String)ois.readObject();
JOptionPane.showMessageDialog(null,stringFromServer); s.close();
}catch(Exception ex){
System.out.println(ex);
}
}
}
Server.java import java.io.*; import java.net.*; public class Server{
public static void main(String args[])
{
try{
ServerSocket ss=new ServerSocket(2222);
System.out.println("Server started");
while(true){
Socket s=ss.accept();
System.out.println("connection request received");
InputStream is=s.getInputStream();
ObjectInputStream ois=new ObjectInputStream(is);
OutputStream os=s.getOutputStream();
ObjectOutputStream oos=new ObjectOutputStream(os);
Read String Object from network String original=(String)ois.readObject();
int length=original.length();
String reverse="";
for(int i=length-1; i>=0; i--){
reverse=reverse+original.charAt(i);
}
if(original.equals(reverse)){
String msg="String is Palindrome";

```
oos.writeObject(msg);
}
else{
String msg="String is not Palindrome";
oos.writeObject(msg);
}
s.close();
}
}catch(Exception ex){
System.out.println(ex);}
}
}//
```

3. Write a JSP page, which will dynamically create a drop down list with all countries along with their regions saved in data base.
DB Name: Database
Table name: Countries(countryName(varChar(50)), regionName(varChar(50)))

```
<%@ page=import="java.sql.*"%>

<%ResultSet rs=null;%>

<HTML>

<HEAD> JSP DROP DOWN LIST</HEAD>

<BODY>

<%

try{

class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

String url="jdbc:odbc:countryDSN";

Connection con=DriverManager.getConnection(url);

Statement st=con.createStatement();

String sql="SELECT *FROM Countries";
ResultSet rs=st.executeQuery(sql);

%>

<h1>Drop Down List</h1>

<select>

<% while(rs.next()){ %>

        <option><%=rs.getString(countryName);

                    rs.getString(regionName);

                %>

       </option>

 <% } %>

 </select>

<%

  }catch(Exception e)
```

```
    {

        System.out.println(e);

    }
%>

</BODY>
</HTML>
```