SOLVED PAST PAPER OF INTERNET PROGRAMMING 2016

Q#2. SHORT ANSWERS:

1. What is the major difference between Java and C++?

A: The differences between the programming languages C++ and Java can be traced to their heritage, as they have different design goals. ... Java is a statically typed object-oriented language that uses a syntax similar (but incompatible) to C++. It includes a documentation system called Javadoc.

- ✔ **C++** was designed for systems and applications programming, extending the C programming language. To this procedural programming language designed for efficient execution, C++ has added support for statically typed object-oriented programming, exception handling,scoped resource management, and generic programming, in particular. It also added a standard library which includes generic containers and algorithms.
- ✔ **Java** was created initially as an interpreter for printing systems but grew to support network computing. It was once used as the base for the "HotJava" thin client system. It relies on a virtual machine to be secure and highly portable. It is bundled with an extensive library designed to provide a complete abstraction of the underlying platform. Java is a statically typed object-oriented language that uses similar (but incompatible) syntax to C++. It was designed from scratch with the goal of being easy to use and accessible to a wider audience. It includes an extensive documentation called Javadoc

The main difference between java and C++ is; Java is platform independent language and it is mainly used for design web based application but C++ is platform dependent language and it is mainly used for design desktop application.

Java	C++
Java does't support Pointer concept	It support pointer concept.
It does't support multiple inheritances.	It support multiple inheritance
Java does not include structures or unions.	It have structure and union concept.
Java includes automatic garbage collection.	C++ requires explicit memory management
Java has method overloading, but no operator overloading.	C++ supports both method overloading and operator overloading.
It is platform independent programming language	It is platform dependent programming language.
It is mainly used for design web based application but also use for develop desktop application.	It is used for design only desktop application like OS, Compiler etc.
Java uses compiler and interpreter both	C++ use only Compiler.
Java is high level programming language in java we write code like simple English language.	C++ is more nearer to hardware then Java

2. Repeated Same Q2 in solved past paper 2015
3. Repeated Same Q3 in solved past paper 2015
4. Repeated Same Q4 in solved past paper 2015

5. How does Java event handling process register the events?

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.
Java Event classes and Listener interfaces

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

Steps to perform Event Handling
Following steps are required to perform event handling:
Register the component with the Listener
Registration Methods
For registering the component with the Listener, many classes provide the registration methods. For example:

- **Button**
- public void addActionListener(ActionListener a){}
- **MenuItem**
- public void addActionListener(ActionListener a){}

TextField

- public void addActionListener(ActionListener a){}
- public void addTextListener(TextListener a){}
- **TextArea**
- public void addTextListener(TextListener a){}

- **Checkbox**
- public void addItemListener(ItemListener a){}
- **Choice**
- public void addItemListener(ItemListener a){}
- **List**
- public void addActionListener(ActionListener a){}
- public void addItemListener(ItemListener a){}

Java Event Handling Code

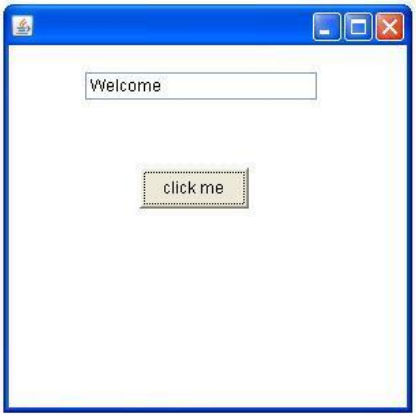
We can put the event handling code into one of the following places:

1. Within class
2. Other class
3. Anonymous class

Java event handling by implementing ActionListener

```
import java.awt.*;
import java.awt.event.*;
class AEvent extends Frame implements ActionListener{
TextField tf;
AEvent(){

//create components
tf=new TextField();
tf.setBounds(60,50,170,20);
Button b=new Button("click me");
b.setBounds(100,120,80,30);

//register listener
b.addActionListener(this);//passing current instance

//add components and set size, layout and visibility
add(b);add(tf);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public void actionPerformed(ActionEvent e){
tf.setText("Welcome");
}
public static void main(String args[]){
new AEvent();
}
}
```

public void setBounds(int xaxis, int yaxis, int width, int height); have been used in the above example that sets the position of the component it may be button, textfield etc.



6. Repeated Same Q4 in solved past paper 2015
7. Repeated Same Q4 in solved past paper 2015

8. What is meant by Server side scripting language?
A: Server-side scripting as it relates to web pages usually refers to PHP code that is executed on the web server before the data is passed to the user's browser. In the case of PHP, all PHP code is executed server-side and no PHP code ever reaches the user. After the PHP code is executed, the information it outputs is embedded in the HTML, which is sent to the viewer's web browser.
One way to see this in action is to open one of your PHP pages in a web browser and then choose the "'View Source" option.
You see the HTML, but no PHP code. The result of the PHP code is there because it is embedded in the HTML on the server before the web page is delivered to the browser.
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. Scripts can be written in any of a number of server-side scripting languages that are available (see below). Server-side scripting is distinguished from client-side scripting where embedded scripts, such as JavaScript, are run client-side in a web browser, but both techniques are often used together.
Server-side scripting is often used to provide a customized interface for the user. These scripts may assemble client characteristics for use in customizing the response based on those characteristics, the user's requirements, access rights, etc. Server-side scripting also enables the website owner to hide the source code that generates the interface.
9. What is mean by mark-up language?
A: A markup language is a set of tags and/or a set of rules for creating tags that can be embedded in digital text to provide additional information about the text in order to facilitate automated processing of it, including editing and formatting for display or printing.
Markup languages are fundamental to displaying documents in web browsers, and they are also employed by every word processing program and by nearly every other program that displays text. However, such languages and their tags are typically hidden from the user

- A markup language is a computer language that uses tags to define elements within a document. It is human-readable, meaning markup files contain standard words, rather than typical programming syntax. Markup languages are designed for the processing, definition and presentation of text. The language specifies code for formatting, both the layout and style, within a text file. The

code used to specify the formatting are called tags. HTML is a an example of a widely known and used markup language.

10. repeated same as in solved past paper 2015

Q#3: LONG QUESTIONS:

1. **What is meant by interface and abstract classes? Write a program which gets student name and roll number by command line arguments and displays them in a message box/prompt box.**

A: Abstract class and interface both are used to achieve abstraction where we can declare the abstract methods. Abstract class and interface both can't be instantiated.
But there are many differences between abstract class and interface that are given below.

Abstract class	Interface
1) Abstract class can have abstract and non-abstract methods.	Interface can have only abstract methods. Since Java 8, it can have default and static methods also.
2) Abstract class doesn't support multiple inheritances.	Interface supports multiple inheritances.
3) Abstract class can have final, non-final, static and non-static variables.	Interface has only static and final variables.
4) Abstract class can provide the implementation of interface.	Interface can't provide the implementation of abstract class.
5) The abstract keyword is used to declare abstract class.	The interface keyword is used to declare interface.

6) Example:
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

Simply, abstract class achieves partial abstraction (0 to 100%) whereas interface achieves fully abstraction (100%).

Example of abstract class and interface in Java

Let's see a simple example where we are using interface and abstract class both.

```
//Creating interface that has 4 methods
interface A{
void a();//bydefault, public and abstract
void b();
void c();
void d();
}
  //Creating abstract class that provides the implementation of one method of A interface
abstract class B implements A{
public void c(){System.out.println("I am C");}
}
  //Creating subclass of abstract class, now we need to provide the implementation of rest of the methods
 class M extends B{
```

```
public void a(){System.out.println("I am a");}
public void b(){System.out.println("I am b");}
public void d(){System.out.println("I am d");}
}
  //Creating a test class that calls the methods of A interface
class Test5{
public static void main(String args[]){
A a=new M();
a.a();
a.b();
a.c();
a.d();
}}
```

Output:

I am a
I am b
I am c
I am d

OR

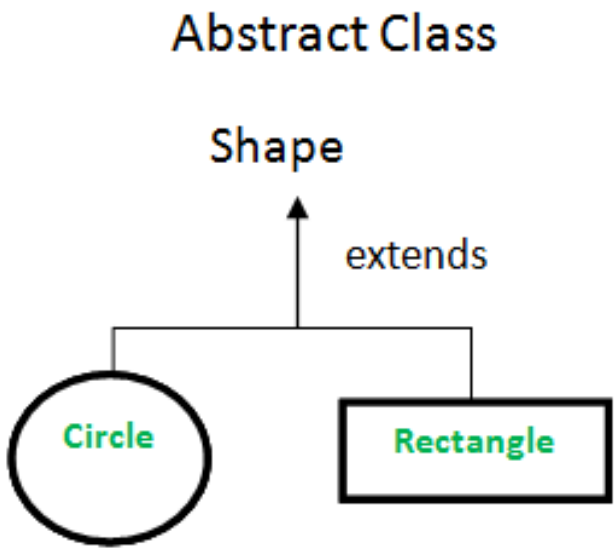
	Abstract Class	Interface
1	An abstract class can extend only one class or one abstract class at a time	An interface can extend any number of interfaces at a time
2	An abstract class can extend another concrete (regular) class or abstract class	An interface can only extend another interface
3	An abstract class can have both abstract and concrete methods	An interface can have only abstract methods
4	In abstract class keyword “abstract” is mandatory to declare a method as an abstract	In an interface keyword “abstract” is optional to declare a method as an abstract
5	An abstract class can have protected and public abstract methods	An interface can have only have public abstract methods
6	An abstract class can have static, final or static final variable with any access specified	interface can only have public static final (constant) variable

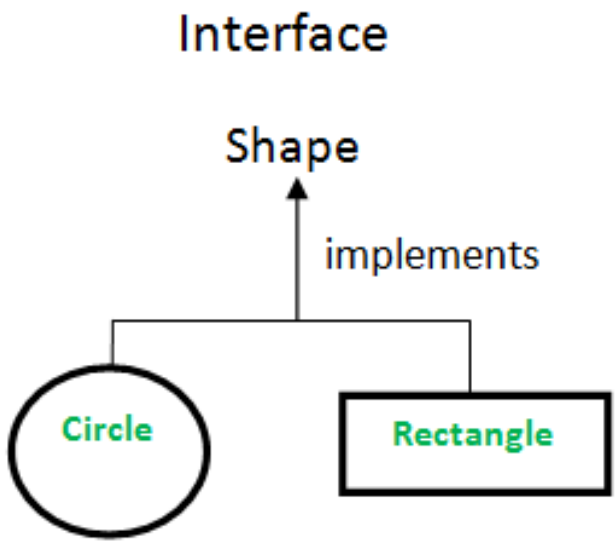
OR

Abstract class vs. Interface

- **Type of methods:** Interface can have only abstract methods. Abstract class can have abstract and non-abstract methods. From Java 8, it can have default and static methods also.
- **Final Variables:** Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.
- **Type of variables:** Abstract class can have final, non-final, static and non-static variables. Interface has only static and final variables.
- **Implementation:** Abstract class can provide the implementation of interface. Interface can’t provide the implementation of abstract class.

- **Inheritance vs. Abstraction:** A Java interface can be implemented using keyword “implements” and abstract class can be extended using keyword “extends”.
- **Multiple implementations:** An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.
- **Accessibility of Data Members:** Members of a Java interface are public by default. A Java abstract class can have class members like private, protected, etc.





For example (pseudo code):

```
// I say all motor vehicles should look like this:
interface MotorVehicle
{
    void run();

    int getFuel();
}

// My team mate complies and writes vehicle looking that way
class Car implements MotorVehicle
{

    int fuel;

    void run()
    {
        print("Wrroooooooom");
    }


    int getFuel()
    {
        return this.fuel;
    }
}
```

Interface example

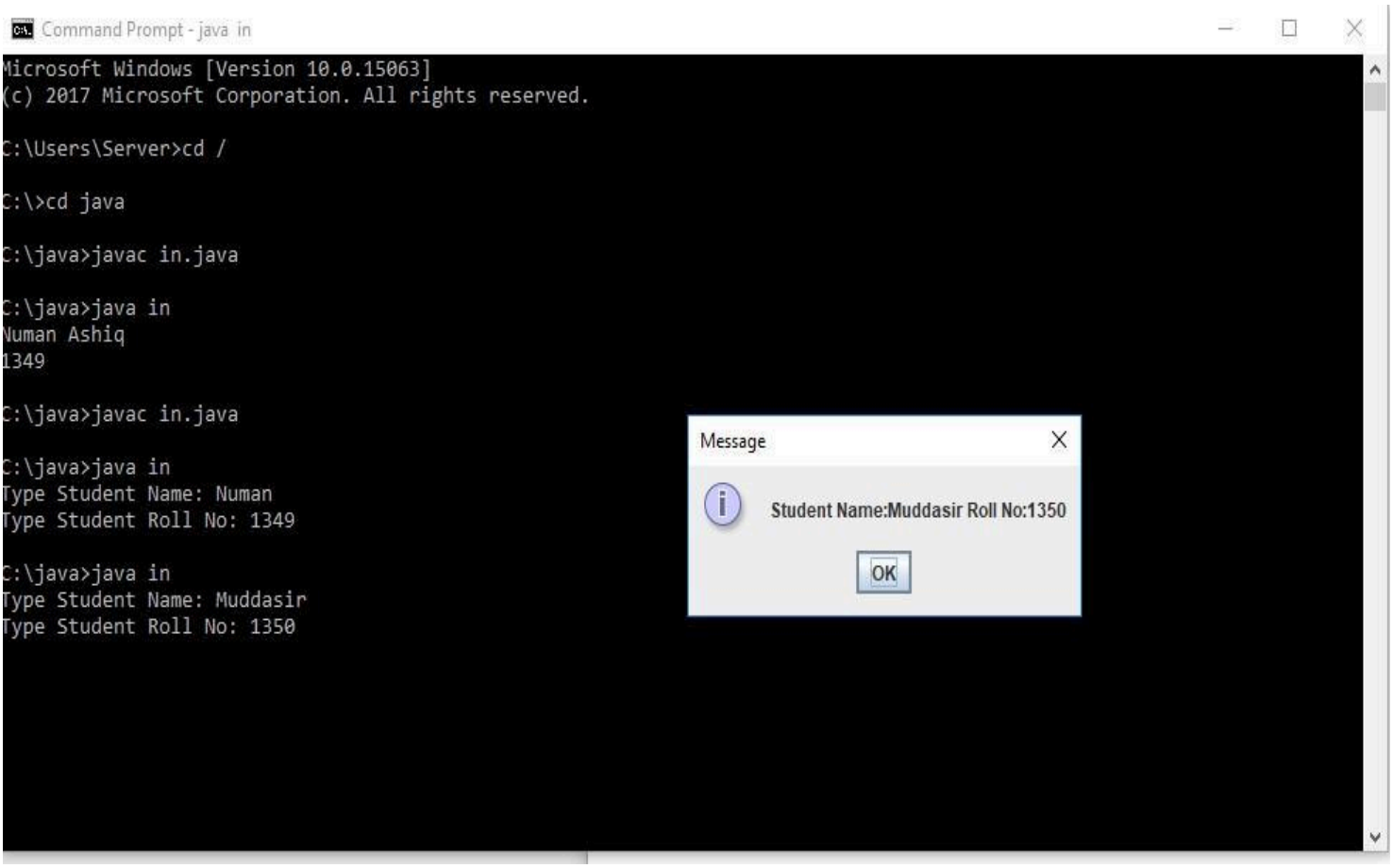
For example:

```
// I say all motor vehicles should look like this:
abstract class MotorVehicle
{

    int fuel;

    // They ALL have fuel, so lets implement this for everybody.
    int getFuel()
    {
         return this.fuel;
    }

    // That can be very different, force them to provide their
    // own implementation.
    abstract void run();
}

// My teammate complies and writes vehicle looking that way
class Car extends MotorVehicle
{
    void run()
    {
        print("Wrroooooooom");
    }
}
```

Abstract Class

Write a program which gets student name and roll number by command line arguments and displays them in a message box/prompt box.

```
import java.util.*;
import javax.swing.*;

public class in{

	public static void main(String[] args){

		Scanner obj = new Scanner(System.in);
		System.out.print("Type Student Name: ");
		String studentName = obj.nextLine();
		System.out.print("Type Student Roll No: ");
		String rollNo = obj.nextLine();

		JOptionPane.showMessageDialog(null, "Student Name:" + studentName + " Roll No:" +
rollNo);


	}
}
```



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).

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.io.InputStream.*;
public class PalindromeServer
{
public static void main(String[] args)
 {
 try{
              /*set the port and tell the server to start accepting sockets*/
              ServerSocket server = new ServerSocket(8000);
                Socket socket = server.accept();
              /* set the In and outputs */
             DataInputStream input = new DataInputStream(socket.getInputStream());
              DataOutputStream output = new DataOutputStream(socket.getOutputStream());
             /*Read in the data from the client*/
              String s = input.readLine();
              output.writeBoolean(isPalindrome(s));
          }
       catch(IOException ex){
       }
   }
   /** Return true if a string is a palindrome */
   public static boolean isPalindrome(String s)
   {
      // Create a new string by eliminating non-alphanumeric chars
```

```
        String s1 = filter(s);
        // Create a new string that is the reversal of s1
        String s2 = reverse(s1);
        // Compare if the reversal is the same as the original string
        return s2.equals(s1);
    }
    /** Create a new string by eliminating non-alphanumeric chars */
    public static String filter(String s)
    {
        // Create a string buffer
        StringBuilder strBuf = new StringBuilder();
        // Examine each char in the string to skip alphanumeric char
        for (int i = 0; i < s.length(); i++)
        {
            if (Character.isLetterOrDigit(s.charAt(i)))
              {
                strBuf.append(s.charAt(i));
            }
        }
        // Return a new filtered string
        return strBuf.toString();
    }
 /** Create a new string by reversing a specified string */
    public static String reverse(String s)
    {
        StringBuilder strBuf = new StringBuilder(s);
        strBuf.reverse(); // Use the reverse method for StringBuilder object
        return strBuf.toString();
    }
}
Here is Client code...
public class PalindromeIgnoreNonAlphanumeric {
  /** Main method */
 public static void main(String[] args) {
    try{
            Socket socket = new Socket("localhost", 8000);
           DataInputStream in = new DataInputStream(socket.getInputStream());
            DataOutputStream out = new DataOutputStream(socket.getOutputStream());
            // Prompt the user to enter a string

            String s = JOptionPane.showInputDialog("Enter a string:");
 out.writeChars(s);
            boolean fromserver = in.readBoolean();
            JOptionPane.showMessageDialog(null, "Your input" + s + "was sent to the server, and the server replied.  Is it a
palindrome? " + fromserver);
        }
    catch(IOException ex){}
  }
}
```

Q3: Write a JSP page, which will dynamically create a drop down list with all smartphones along with their company saved in data base.

DB Names: Database

Table name: Smartphone(model (varChar(15)), company(varChar(15)))

```
<%@page import="java.sql.*"%>
<form name="form" method="post" >
```

```
<b>Select a country:</b>
<select name="countries"><option value=""><---Select---></option>
<%
Class.forName("com.mysql.jdbc.Driver").newInstance();
String connectionURL = "jdbc:mysql://localhost:3306/database";
Connection connection= DriverManager.getConnection(connectionURL, "root", "");
PreparedStatement psmnt = connection.prepareStatement("select * from smartPhones ");
ResultSet results = psmnt.executeQuery();
while(results.next()){
String id = results.getString(1);
String model = results.getString(2);
String company = results.getString(3);
%><option value="<%= model + "-" + company %>"><%=model + "-" + company%></option>
<%} results.close(); psmnt.close(); %>
</select><br>
</form>
```