Q2.1 Describe Whitespaces and Identifiers in Java.

ANSWER:

In Java, a whitespace is a space, tab or newline. Java is a free form language. This means that you do not need to follow any special indentation rules. For instance, any program can be written all on one line or in any other strange way you feel like typing it as long as there is at least one whitespace character between each token that is not already delineated by an operator or separator.

Identifiers are used for class names, method names, and variable names. An identifier may be any descriptive sequence of uppercase and lowercase letters, numbers or the underscore and dollar-sign characters. They must not begin with a number, lest they can be confused with a numeric literal. Since Java is case-sensitive, so ABC is a different identifier than abc.

Q2.2 Why do we need import statement in Java? Give an example.

ANSWER:

Import statement in java allows users to access classes from different packages using the class name instead of prefixing with the full package name.

For example if you are referring to java.util.Date in your class, then instead of referring to java.util.Date where ever the date class is being used, you can just to refer it by Date. So your class would look like this

```
import java.util.Date;

………

……….

Date date = new Date();

System.out.println("Current Date is " + date);
```

Q2.3 Describe the Type Promotion Rules in Java.

ANSWER:

1. First, all byte, short and char values are promoted to int.
2. Then if one operand is a long, the whole expression is promoted to long.
3. If one operand is a float, the entire expression is promoted to float.
4. If any operand is double, the result is double.

Q3.1 Does importing a package imports the subpackages as well? e.g. Does importing com.MyTest.* also import com.MyTest.UnitTests.*?

ANSWER:

No you will have to import the subpackages explicitly. Importing com.MyTest.* will import classes in the package MyTest only. It will not import any class in any of it's subpackage.

Q3.2 Support any two alternatives to inheritance? Give examples.

ANSWER:

Delegation is an alternative to inheritance. Delegation means that you include an instance of another class as an instance variable, and forward messages to the instance. It is often safer than inheritance because it forces you to think about each message you forward, because the instance is of a known class, rather than a new class, and because it doesn't force you to accept all the methods of the super class: you can provide only the methods that really make sense. On the other hand, it makes you write more code, and it is harder to re-use (because it is not a subclass).

Q3.3 Write a program in Java to compute sum of digits of a given number.

ANSWER:

```
void main()

{
int i,sum=0,n;
System.out.print("Enter a number:");
int n =input.nextInt();
while(n!=0)

{
i=n%10;
n=n/10;
sum=sum+i;
}
System.out.println("The sum=",sum);
}
```

Q4.1 What happens when an exception occurs within a method? Explain with an example.

ANSWER:

When an exception occurs within a method, the method creates an exception object and hands it off to the runtime system
– Creating an exception object and handing it to the runtime system is called "throwing an exception"
– Exception object contains information about the error, including its type and the state of the program when the error occurred
The runtime system searches the call stack for a method that contains an exception handler
When an appropriate handler is found, the runtime system passes the exception to the handler
– An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler
– The exception handler chosen is said to catch the exception.

If the runtime system exhaustively searches all the methods on the call stack without finding an appropriate exception handler, the runtime system (and, consequently, the program) terminates and uses the default exception handler

Q4.2 What is synchronization? When should Synchronization be used in Java? Which are the three ways to define synchronized blocks?

ANSWER:

Generally each thread will have its own resource or data to action. In this case there is no possibility for data inconsistency. Sometimes two threads may act on one resource of data and in this case data inconsistency may occur. To overcome this problem java introduces synchronization concept. In synchronization only one thread allowed to operate on the resource when a thread access a synchronized data it calls wait method it is a indication to the thread scheduler not to allow any thread to access the same resource until the first thread job is completed. "Synchronized" is a keyword which can be used to tell through method. We can make a method as synchronized and block as synchronized. We can use synchronized keyword to avoid inconsistency.

Say you have two threads T1 and T2 and both want to access the variable 'a'. Now if T1 changes the value of 'a' from 1 to 2 and then T2 reads it but then T1 changes the value again now from 2 to back 1 what T2 read was a wrong value of 'a'. To avoid this Locks are used. T1 has the lock now until it finishes its work with 'a' then releases the lock. T2 now will gain the lock and work on 'a'.

Java offers three ways to define synchronized blocks.

Synchronized Class Method:

```
class class_name {
  static synchronized type method_name() {
    statement block
  }
}
```

All the statements in the method become the synchronized block, and the class object is the lock.

Synchronized Instance Method:

```
class class_name {
  synchronized type method_name() {
    statement block
  }
}
```

All the statements in the method become the synchronized block, and the instance object is the lock.

Synchronized Statement:

```
class class_name {
  type method_name() {
    synchronized (object) {
      statement block
    }
  }
}
```

All the statements specified in the parentheses of the synchronized statement become the synchronized block, and the object specified in the statement is the lock.

Java applys the synchronization rule by assigning the ownership of the lock's monitor to the threads that are running the synchronized blocks. Here is how it works:

- When a synchronized clock is reached in an execution thread, it will try to gain the ownership of the monitor of the lock object. If another thread owns the lock's monitor, it will wait.
- Once the lock's monitor is free, the waiting thread will become the owner of the lock's monitor, and start to execute the synchronized block.
- Once the synchronized block is executed to the end, the lock's monitor will be freed again.

Note that one program can have many locks and each lock can be associated with many different synchronized blocks. But the synchronization rule only applies between the synchronized block and its associated lock.

For example, the following code defines two synchronized blocks. Both are associated with the same lock, the instance object.

```
class class_name {
  type method_name() {
    synchronized (this) {
      statement block 1
    }
  }
  synchronized type method_name() {
    statement block 2
  }
}
```

Block 1 will never be executed at the same time as block 2.

The following code defines two synchronized blocks. But they are associated with two different locks, one is the class object, and the other is the instance object. Those two synchronized blocks will never wait for each other.

```
class class_name {
  type method_name() {
    synchronized (this) {
      statement block 1
    }
  }
  static synchronized type method_name() {
    statement block 2
  }
}
```

Q5.1 Is it possible to concatenate strings with other types of data? Please illustrate using examples.

ANSWER:

Yes it is possible to concatenate strings with other types of data. Let’s take an example:

```
int age = 9;
String s = "He is " + age + " years old.";
System.out.println(s);
```

Here age is an int rather than another String, but the output produced is the same as before. This is because the int value in age is automatically converted into its string representation within a String object. This string is then concatenated as before. The compiler will convert an operand to its string equivalent whenever the other operand of the + is an instance ofString.

Example:

```
String s = "four: " + 2 + 2;
System.out.println(s);
```

Output:

```
four: 22
```

rather than the

```
four: 4
```

Operator precedence causes the concatenation of “four” with the string equivalent of 2 to take place first. This result is then concatenated with the string equivalent of 2 a second time. To complete the integer addition first, you must use parentheses, like this:

```
String s = "four: " + (2 + 2);
```

Now s contains the string “four: 4”.

Q5.2 What is stream in Java? Classify streams.

ANSWER:

A stream is data that you access in sequence. You could think of it like a train that you watch from a tunnel entrance so that you can just see one car at a time. Or a stream of widgets coming across a conveyor belt requiring you to tighten a screw on each one before it passes by to the next person down the assembly line who has to pound it with a hammer, and so on. Or sticks floating down a river while you watch from a bridge.

Q5.3 Write a program in Java to show how to use the class FileInputStream to read a file named abc.dat. This code should print each byte's value separated with white spaces. Byte values are represented by integers from 0 to 255, and if the read() method returns -1, this indicates the end of the stream.

ANSWER:

```
import java.io.FileInputStream;
import java.io.IOException;
public class ReadingBytes {
public static void main(String[] args) {  FileInputStream myFile = null;  try {
  myFile = new FileInputStream("c:\\abc.dat");  // open the  stream
  boolean eof = false;
   while (!eof) {
    int byteValue = myFile.read();  // read  the stream
    System.out.println(byteValue);
    if (byteValue  == -1){
    eof = true;
    }
    //myFile.close();  // do not do it here!!!
   }
  }catch (IOException e) {
     System.out.println("Could not read file: " + e.toString());
  } finally{
   try{
     if (myFile!=null){
       myFile.close(); // close the stream
     }
   } catch (Exception e1){
    e1.printStackTrace();
   }
  }
 }
}
```

Q6.1 What is collection Interface in Java? Please illustrate with an example.

ANSWER:

A Collection represents a group of objects known as its elements. The Collection interface is used to pass around collections of objects where maximum generality is desired. For example, by convention all general-purpose collection implementations have a constructor that takes a Collection argument. This constructor, known as a conversion constructor, initializes the new collection to contain all of the elements in the specified collection, whatever the given collection's subinterface or implementation type. In other words, it allows you to convert the collection's type.

Suppose, for example, that you have a Collection<String> c, which may be a List, a Set, or another kind of Collection. This idiom creates a new ArrayList (an implementation of the List interface), initially containing all the elements in c.

List<String> list = new ArrayList<String>(c);
The following shows the Collection interface.

```
public interface Collection<E> extends Iterable<E> {
    // Basic operations
    int size();
    boolean isEmpty();
    boolean contains(Object element);
    // optional
    boolean add(E element);
    // optional
    boolean remove(Object element);
    Iterator<E> iterator();

    // Bulk operations
    boolean containsAll(Collection<?> c);
    // optional
    boolean addAll(Collection<? extends E> c);
    // optional
    boolean removeAll(Collection<?> c);
    // optional
    boolean retainAll(Collection<?> c);
    // optional
    void clear();

    // Array operations
    Object[] toArray();
    <T> T[] toArray(T[] a);
}
```

The interface does about what you'd expect given that a Collection represents a group of objects. The interface has methods to tell you how many elements are in the collection (size, isEmpty), to check whether a given object is in the collection (contains), to add and remove an element from the collection (add, remove), and to provide an iterator over the collection (iterator).

The add method is defined generally enough so that it makes sense for collections that allow duplicates as well as those that don't. It guarantees that the Collection will contain the specified element after the call completes, and returns true if the Collection changes as a result of the call. Similarly, the remove method is designed to remove a single instance of the specified element from the Collection, assuming that it contains the element to start with, and to return true if the Collection was modified as a result.

Q6.2 What is Swing in Java? Explain Jlabel and JtextField?

ANSWER:

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check box and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

The Origins of Swing
The Internet Foundation Classes (IFC) were a graphics library for Java originally developed by Netscape Communications Corporation and first released on December 16, 1996. On April 2, 1997, Sun Microsystems and Netscape Communications Corporation announced their intention to incorporate IFC with other technologies to form the Java Foundation Classes. The "Java Foundation Classes" were later renamed "Swing".

Swing introduced a mechanism that allowed the look and feel of every component in an application to be altered without making substantial changes to the application code. The introduction of support for a pluggable look and feel allows Swing components to emulate the appearance of native components while still retaining the benefits of platform independence. Originally distributed as a separately downloadable library, Swing has been included as part of the Java Standard Edition since release 1.2. The Swing classes and components are contained in the javax.swing package hierarchy.

Q7.1 Please list the five steps in the framework of an HTTP transaction.

ANSWER:

1. Connection - A browser (client) opens a connection to a server.
2. Query - The client requests a resource controlled by the server.
3. Processing - The server receives and processes the request.
4. Response - The server sends the requested resource back to the client.
5. Termination - The transaction is done and the connection is closed unless another transaction will take place immediately between the client and server.

Q7.2 What are the general syntax rules for using XHTML?

ANSWER:

- All tags begin with < and ends with >. The tag name is given immediately following the leading <. Make sure the tag is is spelled correctly. Unrecognized tags are ignored by browsers. Any attributes are given following the tag name in the form:

  <tag attribute1 ="value" attribute2 ="value" ... >

- Tag and attribute names are given in lower case. Attributes are always given in the form

  attributeName="value"

  where the value is case sensitive.

- Unrecognized tags and attributes are ignored by browsers.

- Most elements involve open and close tags. Other elements, such as *<br />*, and *<img ... />* (inline image), do not have closing tags and are known as *empty elements*. Note the use of the extra space in front of `/' for empty elements.

- Elements must be *well-formed*. It means no missing begin or end tags and no improper element nesting. For example,

  <p>Learning <strong>XHTML</p></strong>

  overlaps the tags and is not properly nested. Existing browsers may tolerate such ill-formed code. The correct nesting is

  <p>Learning <strong>XHTML</strong></p>

- Attributes can be required or optional and can be given in any order. If an attribute is not given its default value, if any, is used.

- Extra white space and line breaks are allowed between the tag name and attributes, and around the = sign inside an attribute. Line breaks and whitespace within attribute values are also allowed but should be avoided because they may be treated inconsistently by browsers.

- The body element may contain only block-level HTML elements. Freestanding texts (not enclosed in block elements) or inline elements are not allowed directly in the body element.

Q7.3 Give an example to show how to manage line breaks in XHTML.

ANSWER:

- To force a line break, use the <br /> tag.
- To keep two words on the same line use the non-breaking space (entity &nbsp;) instead of a regular SPACE.
- To indicate where a long word can be broken across lines, use the *soft hyphen* (entity &*shy*;) which is rendered as a HYPHEN (-) only at the end of a line. Browsers generally do not break a word that is hyphenated in the source code.

Q8.1 Describe the concept of Page Forwarding. Use an example.

ANSWER:

Web pages sometimes must move to different locations. But visitors may have bookmarked the old location or search engines may still have the old location in their search database. When moving Web pages, it is prudent to leave a *forwarding page* at the original URL, at least temporarily.

A forwarding page may display information and redirect the visitor to the new location automatically. Use the meta tag

```
<meta http-equiv="Refresh" content="8; url=newUrl " />
```

The http-equiv meta tag actually provides an HTTP response header for the page. The above meta element actually give the response header

```
Refresh 8; url=newUrl
```

The effect is to display the page and load the newUrl after 8 seconds. Here is a sample forwarding page

```
<head><title>Page Moved</title>
<meta http-equiv="Refresh" content="8; url=target_url" />
</head><body>
<h3>This Page Has Moved</h3>
<p>New Web location is: ... </p>
<p>You will be forwarded to the new location
automatically.</p>
</body></html>
```

The Refresh response header (meta tag) can be used to refresh a page periodically to send updated information to the end-user. This can be useful for displaying changing information such as sports scores and stock quotes. Simply set the refresh target url to the page itself so the browser will automatically retrieve the page again after the preset time period. This way, the updated page, containing the same meta refresh tag, will be shown to the user.

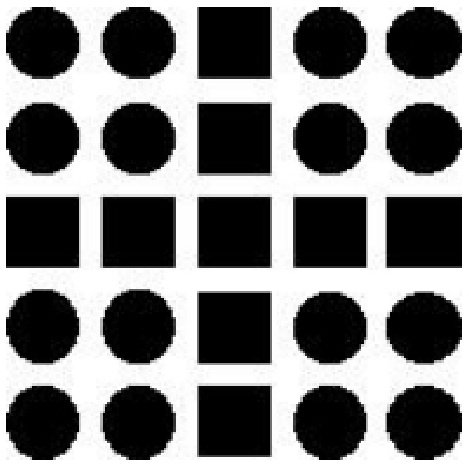
Q8.2 What is gestalt theory in the context of Design and Perception?

ANSWER:

As designers, we have to be aware of many factors involved in communicating with our audience. These factors may sometimes be simple, such as regional and demographic information, while other factors, such as the emotional response of a target audience, may not be quite as easily determined.

Although most designers are not formally trained in psychology, there are a number of fundamental principles designers call upon to explain the working of visual perception. One such theory is the *gestalt* theory, which states that the whole is greater than the sum of its parts. In other words, people perceive an object as a whole unit before they become aware of the individual components it comprises. For example, if a person is looking at a poster of a child on a bicycle, he or she would not notice that the bicycle has 12 spokes on the wheels, or that the child is wearing a button-down sweater with a coarse texture, or that the bicycle is positioned on a grassy path. The viewer would simply see a child on a bicycle first! This theory of basic perception acknowledges the human mind's ability to organize, simplify and unify what it sees. This is how we perceive and understand everything around us.

Given that wholeness is something a mind seeks in order to perceive, it stands to reason that the designer's primary objective is to create unity. Figure below shows how we perceive the cross in the center, before we see anything else.



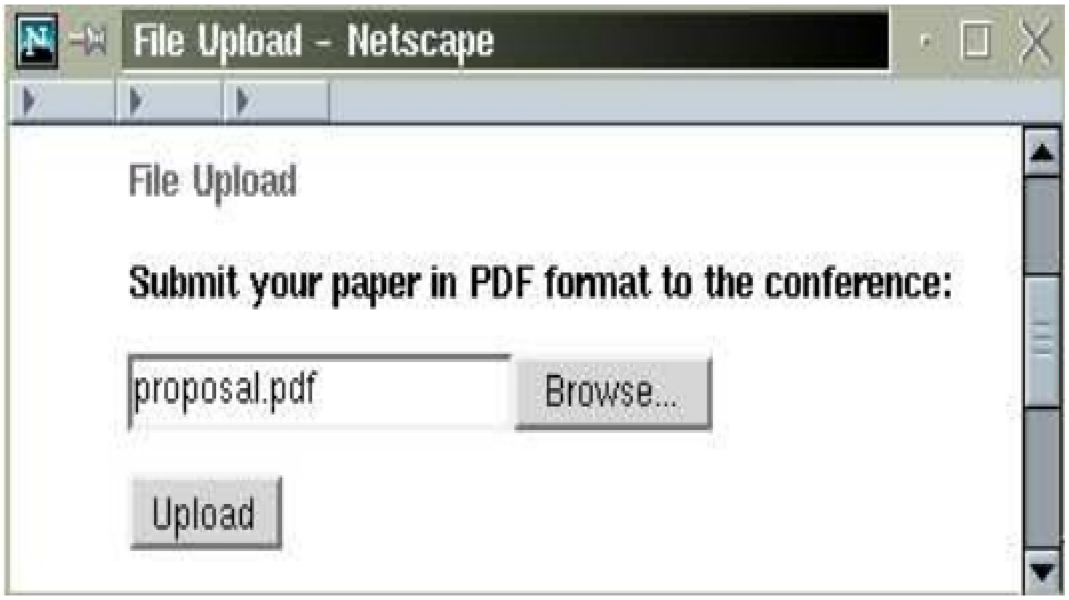
Q9.1 Provide an example of file-upload form where the browse button allows the user to pick a file from the local file system? What are some of the points worth noting about file uploading?

ANSWER:

file upload is done with the file-type input element for example:

```
<form method="post" action="/cgi-bin/receive.cgi"
                        enctype="multipart/form-data">
        <p style="font-size: larger; font-weight: bold;">
                        Submit your paper in PDF format to the conference:</p>
        <p><input type="file" name="paper" accept="application/pdf" />
        </p>
        <p><input type="submit" value="Upload" />
        </p>
</form>
```

Produces the following result:



File Upload - Netscape

File Upload

Submit your paper in PDF format to the conference:

proposal.pdf

Several points worth noting about file uploading are:

- The query method must be post.
- The enctype="multipart/form-data" is needed to specify a data encoding format different from the default application/x-www-form-urlencoded.
- The accept attribute specifies the MIME type for the uploaded file. If accept is not specified, then there is no restriction on the file type.
- Very old browsers may not support form-based file uploading.
- Server-side processing must deal with the multipart/form-data data encoding.

Q9.2 What are the Tasks performed with client-side scripts?

ANSWER:
Tasks performed with client-side scripts include:

- Asking the browser to display information
- Making the Web page different depending on the browser and browser features
- Monitoring user events and specifying reactions
- Generating HTML code for parts of the page
- Modifying a page in response to events
- Checking correctness of user input
- Replacing and updating parts of a page
- Changing the style and position of displayed elements dynamically

<u>TextBooks</u>

1. The Complete Reference Java, Herbert Schildt, TMH, Seventh Edition, 2007

2. An Introduction to Web Design + Programming, Paul S. Wang and Sanda S. Katila, Thomson Course Technology, India Edition, 2008