PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

LAB MANUAL

ACADEMIC YEAR: 2019-20

CLASS: T.E. **SEMESTER: I**

SUBJECT: Skill Development Lab (310246)

Expt. No.	Problem Statement	Revised On
1.	Write Application to make use of collections and generics	15/06/2019
2.	Enhance the system with the help of socket programming use client server architecture to develope chat Server.	15/06/2019
3.	Develope an Application by using JDBC, Multithreading, concurrency, synchronous and asynchronous callbacks, ThreadPools using ExecutorService.	15/06/2019
4.	Transform the above system(assignment 2,3) from command line system to GUI based application	15/06/2019
5.	Download Install and Configure Android Studio on Linux/windows platform.	15/06/2019
6.	Design a mobile app for media player to store data using internal or external storage.	15/06/2019
7.	Design a mobile app using Google Map and GPS to trace the location.	15/06/2019
8.	Mini project	15/06/2019

Subject Coordinator **HOCD**

Ms. Snehal P. Shintre **Dr. R. B. Ingle**

Prerequisites:

Object oriented programming, and basic concepts of data structures.

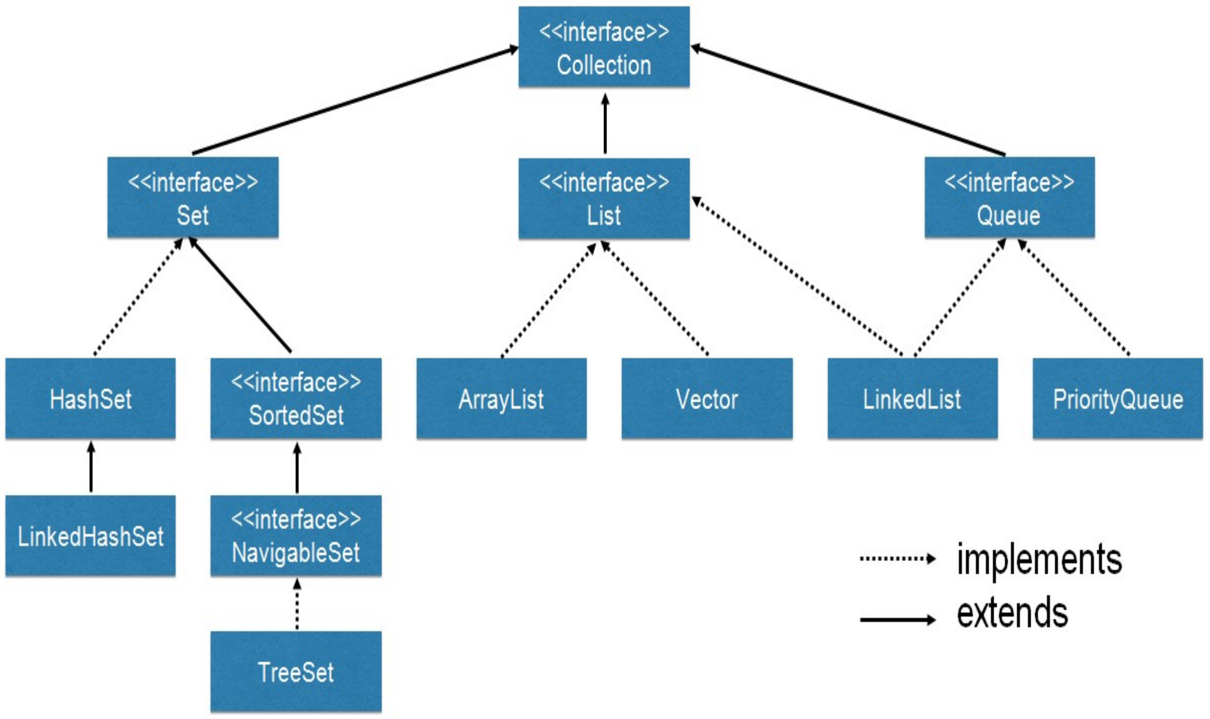
Concepts related Theory:

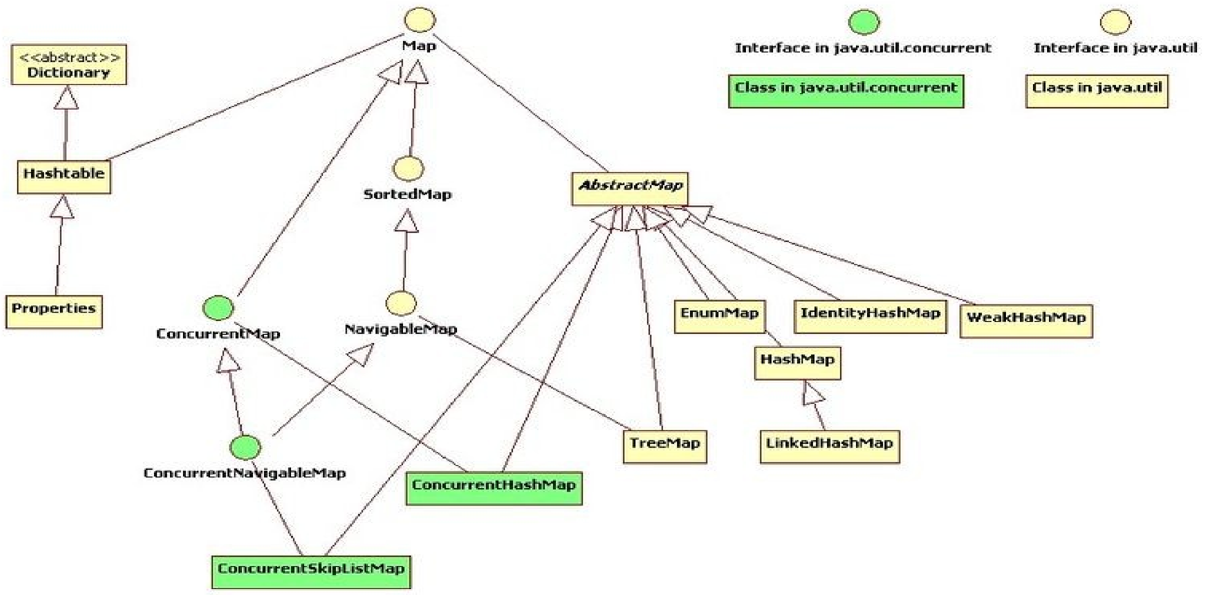
- The data structures provided by the Java utility package are very powerful and perform a wide range of functions. These data structures consist of the following interface and classes −
  - Enumeration
  - BitSet
  - Vector
  - Stack
  - Dictionary
  - Hashtable
  - Properties
- All these classes are now legacy and Java-2 has introduced a new framework called Collections Framework
- All legacy classes and interface were redesign by JDK 5 to support Generics

Java collections framework

The Java collections framework (JCF) is a set of classes and interfaces that implement commonly reusable collection data structures. Although referred to as a framework, it works in a manner of a library. The JCF provides both interfaces that define various collections and classes that implement them.

- A collections framework is a unified architecture for representing and manipulating collections.
- Collection is Group of individual objects represented by single entity
- Advantages
  - Grown able in nature , contains homo and heterogeneous objects
  - Every collections class implemented on DS so we can use readymade methods





Three Types of Collection

There are also three generic types of collection: ordered lists, dictionaries/maps, and sets. Ordered lists allow the programmer to insert items in a certain order and retrieve those items in the same order. An example is a waiting list. Two interfaces are included in the Ordered Lists which are the List Interface and the Queue Interface. Dictionaries/Maps store references to objects with a lookup key to access the object's values. One example of a key is an identification card. The Map Interface is included in the Dictionaries/Maps. Sets are unordered collections that can be iterated and where similar objects are not allowed. The Interface Set is included.

Collection Classes Information:

- The Vector class is similar to a traditional Java array, except that it can grow as necessary to accommodate new elements
- List: Ordered collection of objects, Insertion order is preserve and duplication is allowed (Positional access and insertion is allowed)
- Set: Insertion order is not preserve and duplicate entries are not allowed
- LinkedList: It uses **doubly linked list** to store the elements. It does mmanipulation fast and **act as a list and queue** both because it implements List and Deque interface
- ArrayList: It uses **dynamic array** to store the elements. It does manipulation **slow and act as a list** only because it implements List only
- TreeSet: It is homogeneous Collection of elements and Underline DS is balanced Tree in which Objects are store ascending order(default)
- Queue: It uses when we want group of individual objects before processing ,
- Map: It organize objects which has key-value pair(both objects) and it is collection of entry objects

Generics in Java

Generics are a facility of generic programming that was added to the Java programming language in 2004 within version J2SE 5.0. They were designed to extend Java's type system to allow “a type or method to operate on objects of various types while providing compile-time type safety”. The aspect compile-time type safety was not fully achieved, since it was shown in 2016 that it is not guaranteed in all cases.

- It’s a facility for general programming and designed to extend Java’s type system.
- Facilitates stronger type checks at compile time.
- A Java compiler applies strong type checking to generic code and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find.
- Elimination of casts : without generics requires casting: **List list = new ArrayList(); list.add("hello"); String s = (String) list.get(0);**

- Integer i = (Integer)list.get(0); // Run time error without generics and compile time error with generics
- **String s = list.get(0); // no cast with generics**
- **From JDK 7 , use <> diamond operator to instantiate generic object**
  - Map<String, List<String>> a = new HashMap<String, List<String>>();
  - Replaced by Map<String, List<String>> a = new HashMap<>();
- Example of generic class
  - public class Box<T> { // T stands for "Type" private T t;
  -     public void set(T t) { this.t = t; }
  -     public T get() { return t; } }
  - Box<Integer> integerBox = new Box<Integer>();

Hierarchy and classification

According to Java Language Specification:

- A type variable is an unqualified identifier. Type variables are introduced by generic class declarations, generic interface declarations, generic method declarations, and by generic constructor declarations.
- A class is generic if it declares one or more type variables. These type variables are known as the type parameters of the class. It defines one or more type variables that act as parameters. A generic class declaration defines a set of parameterized types, one for each possible invocation of the type parameter section. All of these parameterized types share the same class at runtime.
- An interface is generic if it declares one or more type variables. These type variables are known as the type parameters of the interface. It defines one or more type variables that act as parameters. A generic interface declaration defines a set of types, one for each possible invocation of the type parameter section. All parameterized types share the same interface at runtime.
- A method is generic if it declares one or more type variables. These type variables are known as the formal type parameters of the method. The form of the formal type parameter list is identical to a type parameter list of a class or interface.

- A constructor can be declared as generic, independently of whether the class that the constructor is declared in is itself generic. A constructor is generic if it declares one or more type variables. These type variables are known as the formal type parameters of the constructor. The form of the formal type parameter list is identical to a type parameter list of a generic class or interface.

The Java collections framework supports generics to specify the type of objects stored in a collection instance.

Conclusion: After successfully completing this assignment, Students will be able develop an application using collection framework and generics.

Review Questions:

1. What is Java Collections Framework? List out some benefits of Collections framework?
2. What is the benefit of Generics in Collections Framework?
3. What are the basic interfaces of Java Collections Framework?
4. Why Collection doesn't extend Cloneable and Serializable interfaces?
5. Why Map interface doesn't extend Collection interface?
6. What is an Iterator?
7. What is difference between Enumeration and Iterator interface?
8. Why there is not method like Iterator.add() to add elements to the collection?
9. Why Iterator don't have a method to get next element directly without moving the cursor?
10. What is different between Iterator and ListIterator?
11. What are different ways to iterate over a list?
12. What do you understand by iterator fail-fast property?
13. What is difference between fail-fast and fail-safe?
14. How to avoid ConcurrentModificationException while iterating a collection?
15. Why there are no concrete implementations of Iterator interface?
16. What is UnsupportedOperationException?
17. How HashMap works in Java?

18. What is the importance of hashCode() and equals() methods?
19. Can we use any class as Map key?
20. What are different Collection views provided by Map interface?
21. What is difference between HashMap and Hashtable?
22. How to decide between HashMap and TreeMap?
23. What are similarities and difference between ArrayList and Vector?
24. What is difference between Array and ArrayList? When will you use Array over ArrayList?
25. What is difference between ArrayList and LinkedList?
26. Which collection classes provide random access of it's elements?
27. What is EnumSet?
28. Which collection classes are thread-safe?
29. What are concurrent Collection Classes?
30. What is BlockingQueue?
31. What is Queue and Stack, list their differences?
32. What is Collections Class?
33. What is Comparable and Comparator interface?
34. What is difference between Comparable and Comparator interface?
35. How can we sort a list of Objects?
36. While passing a Collection as argument to a function, how can we make sure the function will not be able to modify it?
37. How can we create a synchronized collection from given collection?
38. What are common algorithms implemented in Collections Framework?
39. What is Big-O notation? Give some examples?
40. What are best practices related to Java Collections Framework?
41. What is generics in Java ? What are advantages of using Generics?
42. How Generics works in Java ? What is type erasure ?
43. What is Bounded and Unbounded wildcards in Generics ?

44. What is difference between List<? extends T> and List <? super T> ?
45. How to write a generic method which accepts generic argument and return Generic Type?
46. How to write parametrized class in Java using Generics ?
47. Write a program to implement LRU cache using Generics ?
48. Can we use Generics with Array?
49. How can you suppress unchecked warning in Java ?
50. Can you pass List<String> to a method which accepts List<Object>

Prerequisites: Object oriented programming

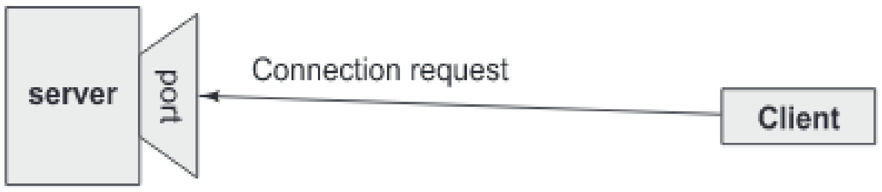
Concepts related Theory:

Introduction:

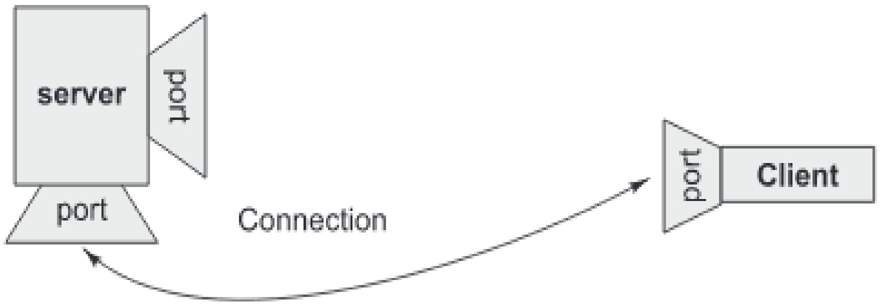
Java creates network applications using sockets. (Others are RMI, web services etc.). Program running on client m/c makes request to a program running on server, involves networking services provided by transport layer. Transport layer has two protocols (TCP, UDP). These use port to map incoming data to a particular process on computer. Multiple services can be run on same time so use ports to distinguish between them.

A socket is a software endpoint that establishes bidirectional communication between a server program and one or more client programs. The socket associates the server program with a specific hardware port on the machine where it runs so any client program anywhere in the network with a socket associated with that same port can communicate with the server program.

Socket provides an interface for programming networks at transport layer. Socket is bound to specific port .Socket act as interface between application and network.



[a]: a client making a connection request to the server



[b]: session established with temporary ports used for two way communication.

A server program typically provides resources to a network of client programs. Client programs send requests to the server program, and the server program responds to the request.

One way to handle requests from more than one client is to make the server program multi-threaded. A multi-threaded server creates a thread for each communication it accepts from a client. A thread is a sequence of instructions that run independently of the program and of any other threads.

Using threads, a multi-threaded server program can accept a connection from a client, start a thread for that communication, and continue listening for requests from other clients.

Java.net Package:
It Contains key classes and interfaces which simplifies client-server programs. **The Classes :** ContentHandler , DatagramPacket , DatagramSocket DatagramSocketImpl ,HttpURLConnection ,InetAddress , MulticastSocket ServerSocket ,Socket, SocketImpl , URL , URLConnection ,URLEncoder URLStreamHandler . **The Interfaces:** ContentHandlerFactory , FileNameMap SocketImplFactory , RLStreamHandlerFactory

TCP/IP Socket Programming:

The steps for creating a simple server program are:

- Open the Server Socket: // 2 methods accept and close

  ServerSocket server = new ServerSocket(PORT);

- Wait for the Client Request:

  Socket client = server.accept();

- Create I/O streams for communicating to the client

  // socket has getInputStream, getOutputStream and close methods

  DataInputStream is = new DataInputStream(client.getInputStream());

  DataOutputStream os = new DataOutputStream(client.getOutputStream());

- Perform communication with client

  Receive from client: String line = is.readLine(); // =(String) is.readUFT();

- Send to client: os.writeBytes("Hello\n");

- Close socket:

  client.close();

Steps for creating client program:

- Create a Socket Object:

  Socket client = new Socket(server, port_id);

- Create I/O streams for communicating with the server.

  is = new DataInputStream(client.getInputStream());

  os = new DataOutputStream(client.getOutputStream());

- Perform I/O or communication with the server:

  Receive data from the server: String line = is.readLine();

- Send data to the server: os.writeBytes("Hello\n");

  //os.writeUTF(line);

- Close the socket when done:

  ```
  client.close();
  ```

UDP Socket Programming

TCP guarantees the delivery of packets and preserves their order on destination. Sometimes these features are not required and since they do not come without performance costs, it would be better to use a lighter transport protocol. This kind of

Service is accomplished by the UDP protocol which conveys datagram packets.

Datagram packets are used to implement a connectionless packet delivery service supported by the UDP protocol. Each message is transferred from source machine to destination based on information contained within that packet. That means, each packet needs to have destination address and each packet might be routed differently, and might arrive in any order. Packet delivery is not guaranteed

It is Lighter protocol which conveys datagram packets. Each packet contains message, length, host, port number. Java supports two classes DatagramSocket, DatagramPacket .

DatagramPacket has various constructors (ex (byte [] buf, int length ,InetAddress addr, int port) and key methods: byte [] getData(), int getLength(); void setData(byte []b) ,void setLegth (int len) .

Most important key methods: void send (DatagramPacket p), void receive (DatagramPacket p)

Applications:

1)Single client-server:

client Programming:

To connect to other machine we need a socket connection. A socket connection means the two machines have information about each other’s network location (IP Address) and TCP port.The java.net.Socket class represents a Socket. To open a socket:

```
Command: Socket socket = new Socket(“127.0.0.1”, 5000)
```

- First argument **–IP address of Server**. (127.0.0.1 is the IP address of localhost, where code will run on single stand-alone machine).
- Second argument **–TCP Port**. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

Communication:

To communicate over a socket connection, streams are used to both input and output the data.

Closing the connection:

The socket connection is closed explicitly once the message to server is sent.

Sever programming:

Establish a Socket Connection

To write a server application two sockets are needed.

- A ServerSocket which waits for the client requests (when a client makes a new Socket())
- A plain old Socket socket to use for communication with the client.

Communication

getOutputStream() method is used to send the output through the socket.

Close the Connection

After finishing,it is important to close the connection by closing the socket as well as input/output streams.

Server application makes a ServerSocket on a specific port which is 5000. This starts our Server listening for client requests coming in for port 5000.

- Then Server makes a new Socket to communicate with the client.

Command: socket = server.accept()

- The accept() method blocks(just sits there) until a client connects to the server.
- Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends “Over”.
- After we’re done we close the connection by closing the socket and the input stream.
- To run the Client and Server application on your machine, compile both of them. Then first run the server application and then run the Client application.

To run on Terminal or Command Prompt

Open two windows one for Server and another for Client

1. First run the Server application as ,

Command: $ java Server

```
Server started
Waiting for a client …
```

2. Then run the Client application on another terminal as,

Command :$ java Client

It will show – Connected and the server accepts the client and shows,

```
Client accepted
```

3. Then you can start typing messages in the Client window. Here is a sample input to the Client

```
Hello

I made my first socket connection

Over
```

Which the Server simultaneously receives and shows,

```
Hello

I made my first socket connection

Over

Closing connection
```

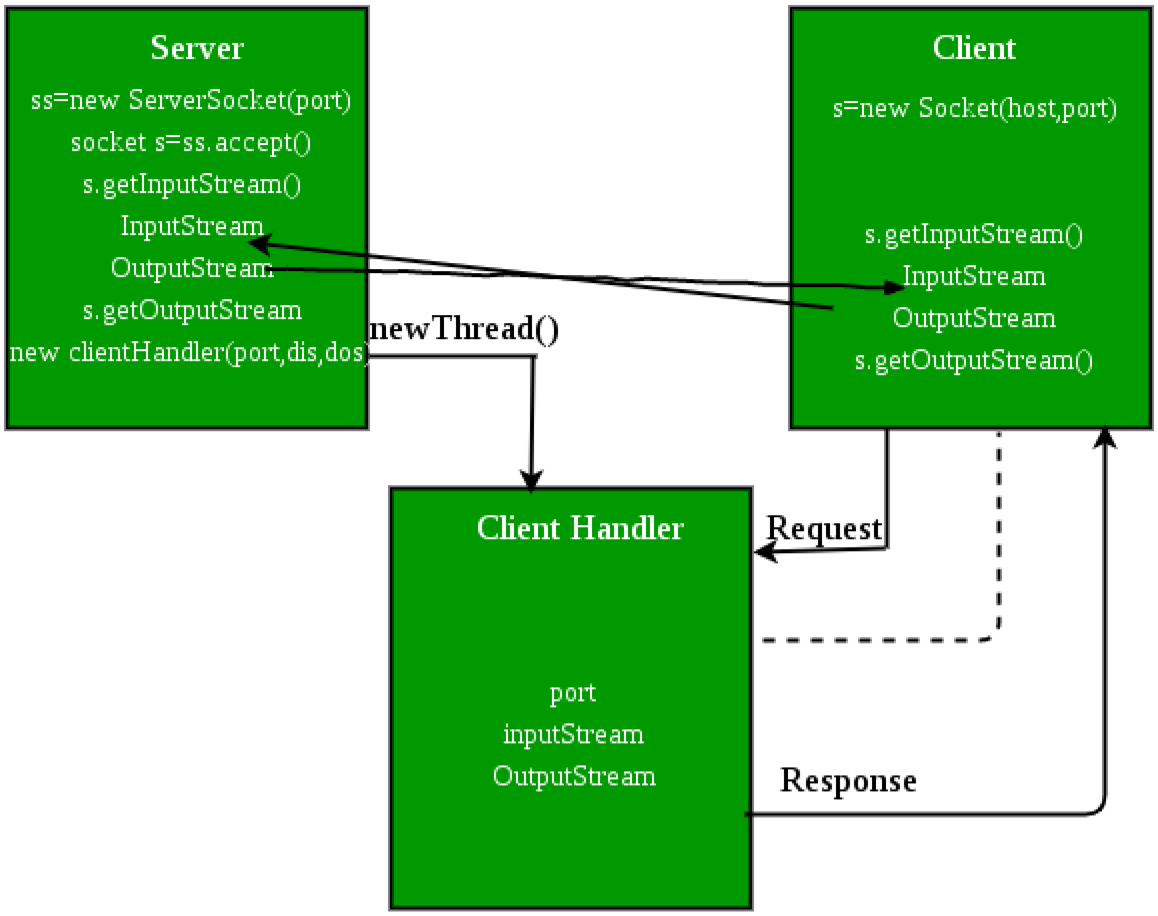
Notice that sending “Over” closes the connection between the Client and the Server just like said before.

If you’re using Eclipse or likes of such-

1. Compile both of them on two different terminals or tabs
2. Run the Server program first
3. Then run the Client program
4. Type messages in the Client Window which will be received and showed by the Server Window simultaneously.
5. Type Over to end.

2)Multiple client-server:

The reason is simple, we don’t want only a single client to connect to server at a particular time but many clients simultaneously. We want our architecture to support multiple clients at the same time. For this reason, we must use threads on server side so that whenever a client request comes, a separate thread can be assigned for handling each request.



We will create two java files,**Server.java** and **Client.java**. Server file contains two classes namely **Server** (public class for creating server) and **ClientHandler** (for handling any client using multithreading). Client file contain only one public class **Client** (for creating a client). Below is the flow diagram of how these three classes interact with each other.

Server Side Programming(Server.java)

- **Server class :** The steps involved on server side are similar to the article Socket Programming in Java with a slight change to create the thread object after obtaining the streams and port number.
  1. **Establishing the Connection:** Server socket object is initialized and inside a while loop a socket object continuously accepts incoming connection.
  2. **Obtaining the Streams:** The inputstream object and outputstream object is extracted from the current requests' socket object.
  3. **Creating a handler object:** After obtaining the streams and port number, a new clientHandler object (the above class) is created with these parameters.

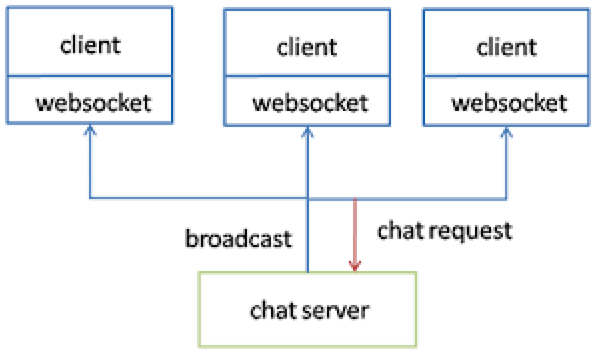
4. **Invoking the start() method :** The start() method is invoked on this newly created thread object.

- **ClientHandler class :** As we will be using separate threads for each request, lets understand the working and implementation of the ClientHandler class extending Threads. An object of this class will be instantiated each time a request comes.
  1. First of all this class extends Thread so that its objects assumes all properties of Threads.
  2. Secondly, the constructor of this class takes three parameters, which can uniquely identify any incoming request, i.e. a **Socket**, a **DataInputStream** to read from and a **DataOutputStream** to write to. Whenever we receive any request of client, the server extracts its port number, the DataInputStream object and DataOutputStream object and creates a new thread object of this class and invokes start() method on it.

     Note : Every request will always have a triplet of socket, input stream and output stream. This ensures that each object of this class writes on one specific stream rather than on multiple streams.
  3. Inside the **run()** method of this class, it performs three operations: request the user to specify whether time or date needed, read the answer from input stream object and accordingly write the output on the output stream object.

3)Chat Server:

In the above model, a simple date time server was created which handled multiple user requests at the same time using threading. It explains the basic concepts of threading in network programming. The same concepts can be used with very slight modification to extend the above idea and create a chatting application similar to facebook messenger, whatsapp etc.



The following steps covers the implementation of such an application with a detailed explanation, limitations, and their solutions.

Server class : The main server implementation is easy and similar to the previous article. The following points will help understand Server implementation :

1. The server runs an infinite loop to keep accepting incoming requests.
2. When a request comes, it assigns a new thread to handle the communication part.
3. The sever also stores the client name into a vector, to keep a track of connected devices. The vector stores the thread object corresponding to the current request. The helper class uses this vector to find the name of recipient to which message is to be delivered. As this vector holds all the streams, handler class can use it to successfully deliver messages to specific clients.
4. Invoke the start() method.

   ClientHandler class : Similar to previous article, we create a helper class for handling various requests. This time, along with the socket and streams, we introduce a name variable. This will hold the name of the client that is connected to the server. The following points will help understand ClientHandler implementation :

1. Whenever the handler receives any string, it breaks it into the message and recipient part. It uses Stringtokenizer for this purpose with '#' as the delimiter. Here it is assumed that the string is always of the format:

   Command: message # recipient

2. It then searches for the name of recipient in the connected clients list, stored as a vector in the server. If it finds the recipients name in the clients list, it forwards the message on its output stream with the name of the sender prefixed to the message.

Limitations:
Although the above implementation of server manages to handle most of the scenarios, there are some shortcomings in the approach defined above.

- One clear observation from above programs is that **if the number of clients grew large, the searching time would increase** in the handler class. To avoid this increase, two hash maps can be used. One with name as the key, and index in active list as the value. Another with index as key, and associated handler object as value. This way, we can quickly look up the two hashmaps for matching recipient. It is left to the readers to implement this hack to increase efficiency of the implementation.
- Another thing to notice is that this implementation **doesn't work well when users disconnect from the server**. A lot of errors would be thrown because disconnection is not handled in this implementation. It can easily be implemented as in previous basic TCP examples. It is also left for the reader to implement this feature in the program.

Conclusion:

Student are able to implement chat server using socket programming in Java

Review Questions:

1. Explain the client-server architecture via a simple example.
2. Explain the TCP/IP stacks.
3. Briefly explain the TCP and UDP protocols and the difference between the two protocols.
4. What is port? List some well-known ports and explain the applications associated with them.
5. Discuss the process of creation of server and client sockets with exceptions handled explicitly with a suitable example.
6. What Is Socket?
7. What Does A Socket Consists Of?
8. What Are The Seven Layers(osi Model) Of Networking?
9. What Are Some Advantages And Disadvantages Of Java Sockets?
10. How Do I Open A Socket?
11. How Do I Create An Input Stream?
12. How Do I Create An Output Stream?
13. How Do I Close Sockets?
14. Explain Data Transfer Over Connected Sockets - Send() And Recv()?
15. Explain Connection Establishment By Server - Accept()?
16. How To Make A Socket A Listen-only Connection Endpoint - Listen()?
17. How Sockets Can Be Used To Write Client-server Applications Using A Connection-oriented Client-server Technique?
18. How To Disposing Of A Socket?
19. What Is Socket Programming?
20. What is the basic concept behind
21. InputStream and OutputStream in Java?
22. What are the functions of the ServerSocket() and the accept() methods?
23. When would you prefer to have a concurrent server?
24. What are the functions of the DatagramPacket and DatagramSocket classes?

Revised on: 15-02-2019

ASSINGMENT NO.	**3**
TITLE	**Application development using JDBC and concurrency**
PROBLEM STATEMENT /DEFINITION	Develop an application by using JDBC, Multithreading, concurrency, synchronous and asynchronous callbacks, ThreadPools using ExecutorService.
OBJECTIVE	1.To learn database connectivity 2.To learn concurrency
OUTCOME	Student should be able to implement 1.All types of JDBC drivers 2.Concurrency in their application
S/W PACKAGES AND HARDWARE APPARATUS USED	Fedora linux, JDK, I3/I5
REFERENCES	“JAVA complete reference” book
INSTRUCTIONS FOR WRITING JOURNAL	• Title • Problem Definition • Objectives: Intention behind study • Software & Hardware requirements • Explanation of the assignment (Theory) • Class Diagram/UML diagram • Algorithm /Steps performed • Installing / Developing the system/program • Command listing & test results • Conclusion

Prerequisites: Students should know database related basic concepts

Concepts related Theory:

Java JDBC is a Java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.

Following are the basic steps in a JDBC application:

1. Import the package.

e.g. import java.sql.*;

2. Load and Register the Driver.
Load: The jdbc driver used for connection should be available in your system. If you are using Eclipse IDE, then you have to provide link in properties while if you are running your jdbc code on terminal, then you have to provide classpath in .bashrc file.

Register: In jdbc code, we need to register a driver for use. A method forName() is provided for the same purpose.
e.g. Class.forName("com.mysql.jdbc.Driver");

3. Establish a Connection to the database
Create a Connection object. Provide URL (with database port number, database name), user name and password.
e.g. Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/databasename","username","userpassword");

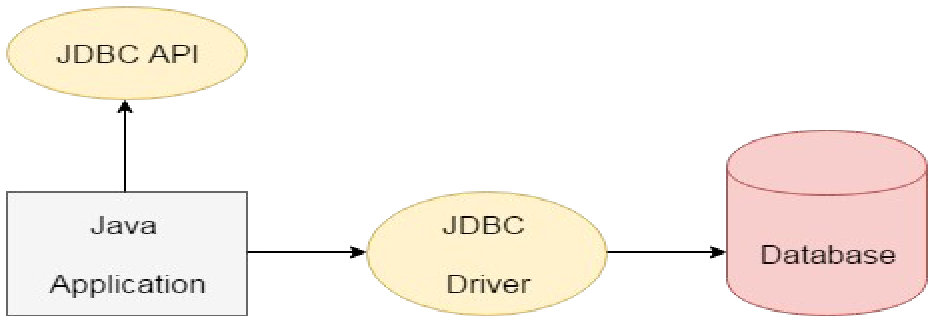
4. Create a Statement object from the Connection.
Statement object is used for executing queries on database.
e.g. Statement st=con.createStatement();

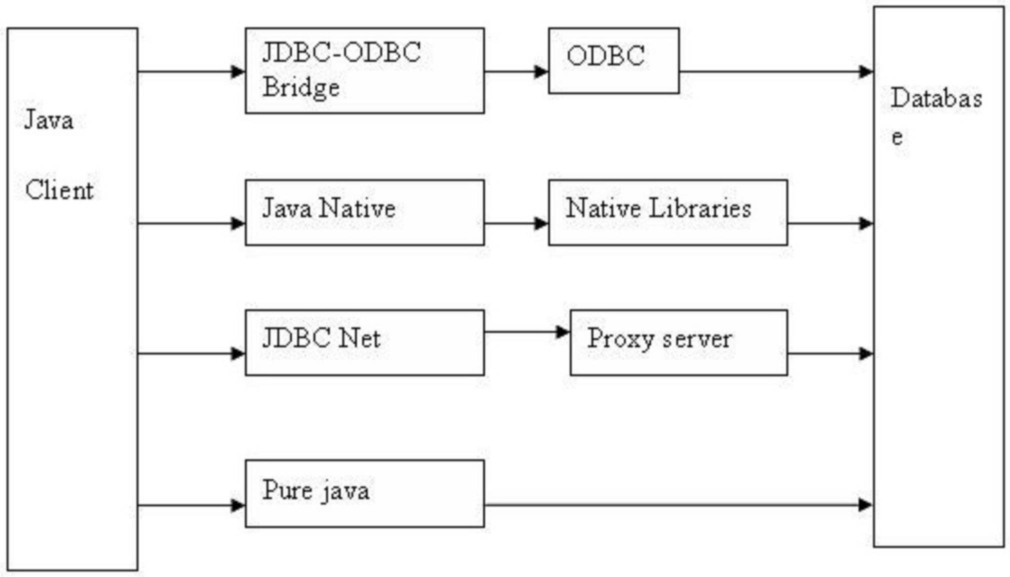
5. Use the Statement object to execute query.
If we are fetching data from database, then we need to define ResultSet object. We can also perform other operations like insert, update, delete on database table.
e.g.
ResultSet rs=st.executeQuery("select * from student");
int i=st.executeUpdate("insert into student values(1201,'se','vjti','mumbai')");

6. Process Result.
If we are fetching data from the database, we can get it from ResultSet object. We can process this data as per our requirement.

7. Close/terminate the objects.
e.g.
rs.close();
st.close();
con.close();

JDBC Drivers:





What is type 1 driver in JDBC?

This is the oldest JDBC driver, mostly used to connect database like MS Access from Microsoft Windows operating system. Type 1 JDBC driver actually translates JDBC calls into ODBC (Object Database connectivity) calls, which in turn connects to database. Since it acts as bridge between JDBC and ODBC, it is also known as **JDBC ODBC bridge driver**. This driver had very poor performance because of several layers of translation which took place before your program connects to database. It has also less portable because it relies on ODBC driver to connect to database which is platform dependent. It is now obsolete and only used for development and testing, I guess Java 7 even removed this driver from JDK.

What is type 2 driver in JDBC?

This was the second JDBC driver introduced by Java after Type 1, hence it known as type 2. In this driver, performance was improved by reducing communication layer. Instead of talking to ODBC driver, JDBC driver directly talks to DB client using native API. That's why its also known as native API or partly Java driver. Since it required native API to connect to DB client it is also less portable and platform dependent. If native library e.g. ocijdbc11.dll, which is required to connect Oracle 11g database is not present in client machine then you will get java.lang.UnsatisfiedLinkError: no dll in java.library.path error. Performance of type 2 driver is slightly better than type 1 JDBC driver.

What is type 3 driver in JDBC?
This was the third JDBC driver introduced by Java, hence known as type 3. It was very different than type 1 and type 2 JDBC driver in sense that it was completely written in Java as opposed to previous two drivers which were not written in Java. That's why this is also known as all Java driver. This driver uses 3 tier approach i.e. client, server and database. So you have a Java client talking to a Java server and Java Server talking to database. Java client and server talk to each other using net protocol hence this type of JDBC driver is also known as Net protocol JDBC driver. This driver never gained popularity because database vendor was reluctant to rewrite their existing native library which was mainly in C and C++

What is type 4 JDBC driver?
This is the driver you are most likely using to connect to modern database like Oracle, SQL Server, MySQL, SQLLite and PostgreSQL. This driver is implemented in Java and directly speaks to database using its native protocol. This driver includes all database call in one JAR file, which makes it very easy to use. All you need to do to connect a database from Java program is to include JAR file of relevant JDBC driver. Because of light weight, this is also known as thin JDBC driver. Since this driver is also written in pure Java, its portable across all platform, which means you can use same JAR file to connect to MySQL even if your Java program is running on Windows, Linux or Solaris. Performance of this type of JDBC driver is also best among all of them because database vendor liked this type and all enhancement they make they also port for type 4 drivers.

Multi-threading in Java:

Different phases in Thread Life Cycle:
Ans: Newborn: New thread is created.
Running: Thread is running on processor core.
Runnable: Thread is waiting for the access of processor core.
Blocked: Thread is suspended.
Dead: Execution of thread is stopped.

In Java, there are two ways of creating threads:

1. By implementing interface Runnable

2. By extending class Thread

Thread Pool in Java:

What is Threadpool?
Threadpool is a concept in Java. It refers to the collection of threads i.e. a group of fixed size of threads.

How it works?
A thread is taken (i.e. pulled) from thread pool and task is allocated to it. Similarly, other threads are taken from thread pool and tasks are allocated to them. When task is completed, thread is returned to the thread pool. A returned thread in thread pool can be pulled back again and can be allocated a new task.

Suppose there are three threads in a thread pool and five tasks.
(First Thread=First Task) First thread will be allocated first task.
(Second Thread=Second Task) Second thread will be allocated second task.
(Third Thread=Third Task) Third thread will be allocated third task.

Once the first or second or third thread will be free i.e. completed task; it will return to thread pool and it will be allocated fourth task.

Again whoever thread gets free will return back to thread pool and will be allocated fifth task.

Advantage of Thread Pool:
Thread Pool reuses the threads. That's why, it reduces the time for creating new threads.
Java Thread Pool can be used with Servlet or JSP.

Conclusion: Students should implement the JDBC drivers successfully

Review Questions:

- How to connect Java application with Oracle and Mysql database using JDBC?
- What is the difference between Statement and PreparedStatement interface?
- How to print total numbers of tables and views of a database using JDBC ?
- What are the types of JDBC drivers?
- What are the basic steps in jdbc starting from connection with database to the execution of SQL queries?
- Which package should be imported in JDBC Java program?

- What is port number used by MySQL database?
- Explain some basic SQL queries with their syntaxes?
- What are the different phases in Thread Life Cycle?
- What are the two ways of creating threads in Java?
- What is Thread Pool concept in Java?
- What is the advantage of Thread Pool?
- What is ExecutorService? How it is used to create pool of threads?

Prerequisites:

Object oriented programming, and core Java.

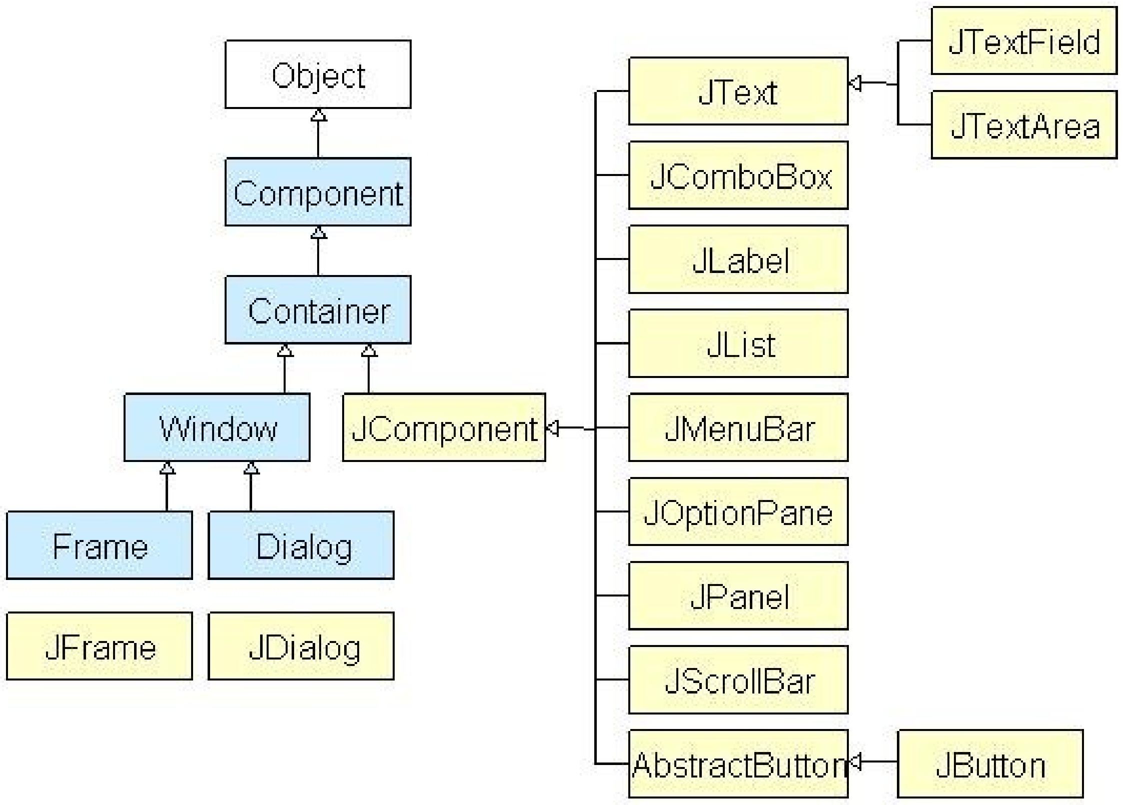
Concepts related Theory:

Java Swing is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Java Swing class Hierarchy Diagram



All components in swing are JComponent which can be added to container classes.

Building a GUI application

1. Identify user requirement for GUI.

2. Map the requirements to Swing components

3. Group UI components.

4. Define UI structure.

5. Select suitable events to handle.

6. Link events to execution logic.

Container class :

Container classes are classes that can have other components on it. So for creating a GUI, we need at least one Container object Three types of containers

4. Panel : It is a pure container and is not a window in itself. The sole purpose of a Panel is to organize the components on to a window.
5. Frame : It is a fully functioning window with its own title and icons.
6. Dialog : It can be thought of as a pop-up window that pops out when message has to be displayed. It is not a fully functioning window like the Frame.

Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below.

Method	**Description**
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Java JButton

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

Commonly used Constructors:

Constructor	**Description**
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.

JButton(Icon i)	It creates a button with the specified icon object.

Commonly used Methods of AbstractButton class:

Methods	**Description**
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

Java JLabel

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

Java JTextField

Methods	**Description**

JText Field()
JTextField(String)
JText Field(String,int)
JTextField(int)
	Creates a text field. When present, the int argument specifies the desired width in columns. The String argument contains the field’s initial text.

void setText(String) String getText()	Sets or obtains the text displayed by the text field.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

Java JTextArea

The object of a JTextArea class is a multi line region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class

Java JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

Method	Description
JCheckBox(Action)	Create a JCheckBox instance. The string

JCheckBox(String)
 	
JCheckBox(String, boolean)
 	
JCheckBox(Icon)
 	
JCheckBox(Icon, boolean)
 	
JCheckBox(String, Icon)
 	
JCheckBox(String, Icon, boolean)
 	
JCheckBox()
	argument specifies the text, if any, that the checkbox should display. Similarly, the icon argument specifies the image that should be used instead of the default check box image. Specifying the Boolean argument as true initializes the check box to be selected . If the boolean argument is absent or false, then the checkbox is initially unselected.

Java JRadioButton

The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz. It should be added in ButtonGroup to select one radio button only.

Method	Description
JRadioButton(Action) JRadioButton (String) JRadioButton (String, boolean) JRadioButton (Icon) JRadioButton (Icon, boolean) JRadioButton (String, Icon)	Create a JRadioButton instance. The string argument specifies the text, if any, that the radiobutton should display. Similarly, the icon argument specifies the image that should be used instead of the default radiobutton image. Specifying the Boolean argument as true initializes the radiobutton to be selected . If the boolean argument is absent or false, then the radiobutton is initially unselected.

JRadioButton (String, Icon, boolean) JRadioButton ()	

Java JComboBox

A component that combines a button or editable field and a drop-down list. The user can select a value from the drop-down list, which appears at the user's request. If you make the combo box editable, then the combo box includes an editable field into which the user can type a value.

Menu in Java Swing

Menus in Swing are UI components which may contain other UI components.

1. A MenuBar acts as a container for multiple menu items.

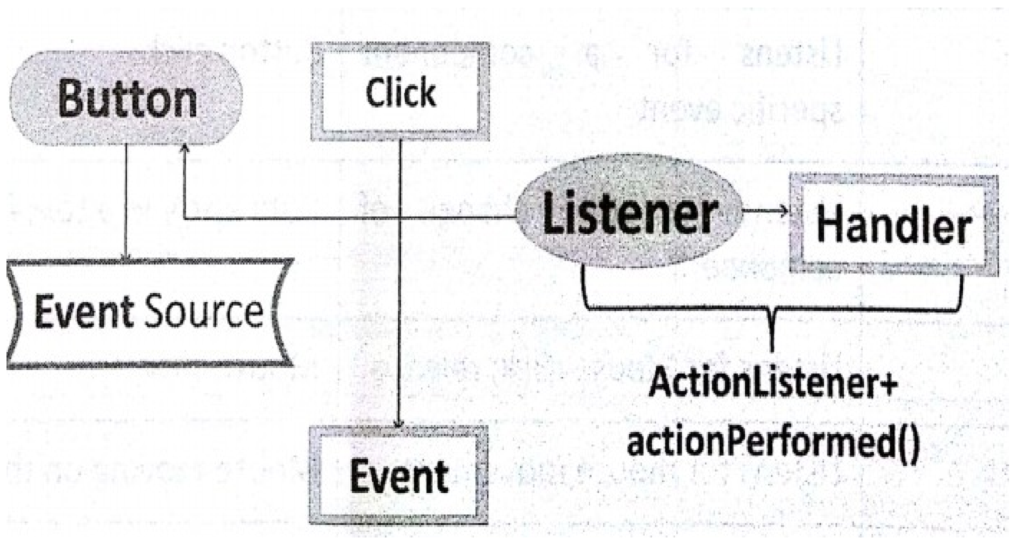
2. MenuItems can be a Check Box or Radio Button.

3. Events are associated with individual MenuItem.

\

Event Listener

The entire process of event delegation model in Swing:



When button is clicked, it creates an Action Event object and invokes actionPerformed(ActionEvent)method of ActionListener interface. All that is needed is to write code inside the actionPerformed method by implementing the ActionListener interface. Event source needs a reference to the object of event handler so that it can call its method.

Event Listeners

Listener	Purpose
ActionListener	Listens for a component specific event.
ChangeListener	Listens for state change of component.
MouseListener	Listens for Mouse click, release
MouseMotionListener	Listens for Mouse movement.
KeyListener	Listens for Keyboard activity.
WindowListener	Listens for window handling
ItemListener	Listens for state change of an item.

AdjustmentListener	Listens for scrollbar movement.
WindowFocusListener	Listens for window activation

Conclusion: After successfully completing this assignment, Students will be able develop an application more user friendly.

Review Questions:

2. What is Swing?
3. What is a container class?
4. What are differences between Swing and AWT?
5. What Is Lightweight Component?
6. Why Swing components are called lightweight components?
7. Which package has light weight components?
8. What is the design pattern that Java uses for all Swing components ?
9. What Is An Event In Swing?
10. What Is An Event Handler In Swing?
11. What advantage do Java's layout managers provide over traditional windowing systems ?
12. What Is A Layout Manager?
13. Which Containers Use A Border Layout As Their Default Layout In Swing?
14. How are the elements of a GridBagLayout organized?
15. What Is The Preferred Size Of A Component?
16. What Method Is Used To Specify A Container's Layout?
17. Which Containers Use A Flow layout As Their Default Layout?
18. Which Method Of The Component Class Is Used To Set The Position And Size Of A Component?
19. Which Method Is Used By The Applet To Recognize The Height And Width?
20. When We Should Go For Codebase In Applet?
21. What Is The Lifecycle Of An Applet?
22. Which Method Is Used For Setting Security In Applets?
23. What Is An Event And What Are The Models Available For Event Handling?
24. What Are the Advantages of the Event-delegation Model Over the Event-inheritance Model?

25. Give Us the Name of the List Layout managers In Java?

26. Which Swing methods are thread-safe ?

27. Name three Component subclasses that support painting?

Revised on: 15-06-2019

ASSINGMENT NO.	**5**
TITLE	**Android studio installation**
PROBLEM STATEMENT / DEFINITION	To install Android Studio on Fedora Linux and develop the android application as per the requirement
OBJECTIVE	1.To learn Android studio installation 2.To learn to develop an android application
OUTCOME	Student should be able to 1.Install the Android studio 2.Develop an application as per requirement
S/W PACKAGES AND **HARDWARE APPARATUS USED**	Fedora Linux, Android studio, Oracle Java JDK 8**.**
REFERENCES	
INSTRUCTIONS FOR **WRITING JOURNAL**	• Title • Problem Definition • Objectives: Intention behind study • Software & Hardware requirements • Explanation of the assignment (Theory) • Class Diagram/UML diagram • Algorithm /Steps performed • Installing / Developing the system/program • Command listing & test results • Conclusion

Prerequisites: Students should know programming related concepts

Concepts related Android Studio:

Android Studio: Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.

Use of Android Studio: It has a strong editor tool for developing creative UI and emulators for different versions to test and simulate sensors without having actual Android devices. It also has a very useful Gradle plugin using which you can create application files (apks) with different configurations. Moreover it makes exporting and uploading apk on playstore easy with a single click. It also has ANT build if you prefer that. In the recent updates Android studio has brought instant run which makes testing even faster and easier.

What is difference between Android Studio and Eclipse?

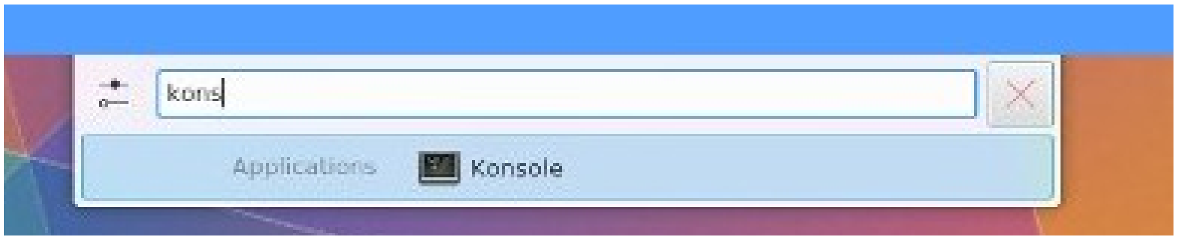
It supports gradle (this allows you to really have control over the build, create different application flavors, different signing configurations and so on). In the layout view you have the option to view both the actual layout and the xml at the same time, while in Eclipse you must choose between the two tabs.

Can you still use eclipse with android?

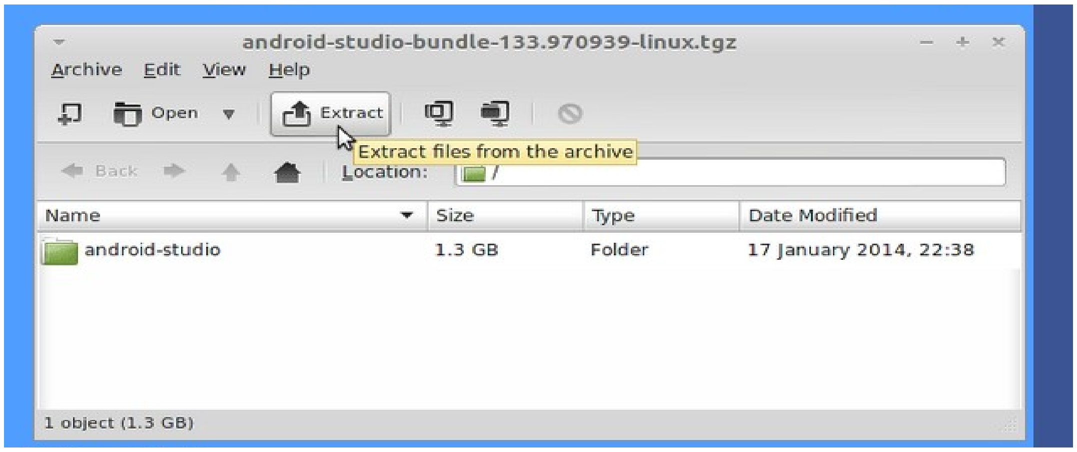
In fact, you can build an Android application with an IDE at all.With the release of Android Studio and the Android Gradle plugin, everything is shifting towards Gradle dependencies. Also be aware that Google is not going to continue supporting Eclipse development

Installation steps:

- First Open a Shell terminal emulator window Start Typing ‘term’ on Desktop(press ‘enter’ to execute command)



- Download Android Studio IDE for Fedora Linux.
- Double-Click/Right-Click on Archive and Extract into /tmp



Or from shell:

```
unzip -d /tmp/ ~/Downloads/android-studio*.zip
```

- **Install required packages**

Type in shell:

```
sudo dnf install compat-libstdc++-296.i686 compat-libstdc++-33.i686 \
compat-libstdc++-33.x86_64 ncurses-libs.i686 zlib.i686 \
bzip2-libs.i686 glibc.i686 glibc-devel.i686 libstdc++.i686 \
zlib-devel.i686 ncurses-devel.i686 libX11-devel.i686 \
libXrender.i686 libXrandr.i686
```

- Install Recommended Oracle Java JDK 8 on Linux Fedora.
- After to Relocate Android Studio IDE.

Set the SuperUser as holder(in shell):

```
Sudo chown -R root:root /tmp/android-studio
```

Then Switch the contents with(in shell):

```
sudo mv /tmp/android-studio /opt\
```

Finally End the SuperUser Session(in shell):

```
exit
```

- Set the Android SDK tools Path

Edit the User Bash configuration file(in shell):

```
nano ~/.bashrc
```

Append(in shell):

```
export PATH=$PATH:/home/[user]/Android/Sdk/tools:/home/[user]/Android/Sdk/platform-tools:/home/[user]/Android/Sdk/tools/bin
```

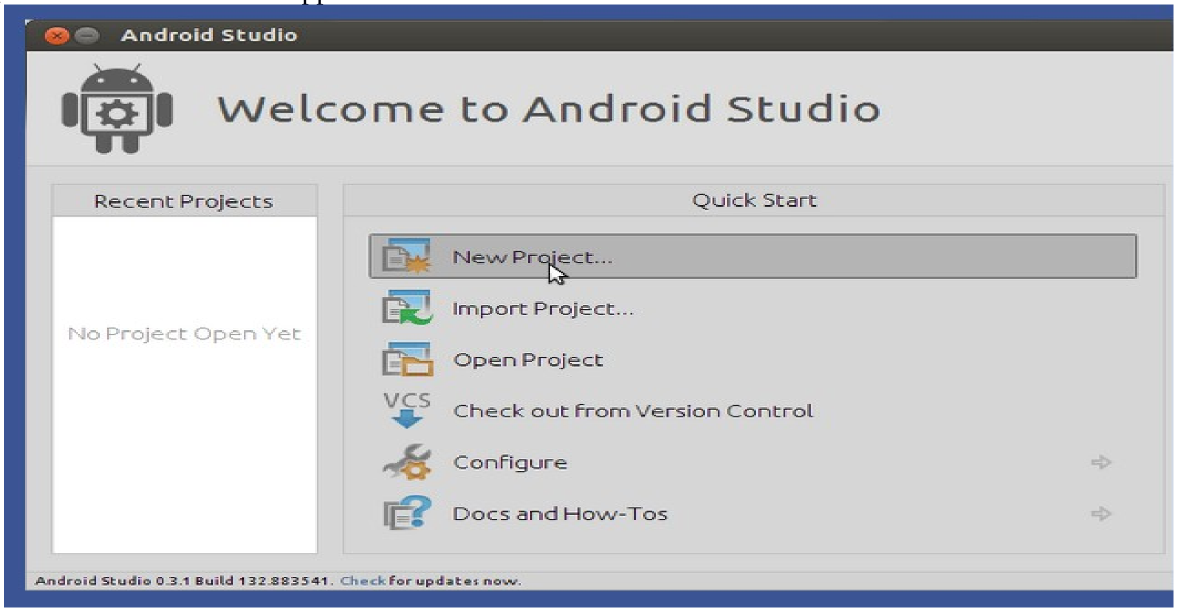
Just Replace [user] with Your User name **Ctrl+x** to Save & Exit from nano editor :)
Load the New Path simply with command : bash

This Setup will be useful after the First launch of Android Studio IDE and the Installation of Android SDK Tools

- Finally to start Android Studio IDE from shell:

```
cd /opt/android-studio
bin/studio.sh
```

- Quick start with Android Apps on Android Studio



Conclusion: Students should install Android Studio and develop an android application

Review Questions:

1. How to implement the project?

2. How to make an app and put it on play store?

3. How to make an app for sell?

Revised on: 15-06-2019

ASSINGMENT NO.	**6**
TITLE	**Media player in android**
PROBLEM STATEMENT / DEFINITION	Design a mobile app for media player to load data from internal or external storage.
OBJECTIVE	• To understand design and development in Android studio • To learn memory management in android
S/W PACKAGES AND HARDWARE APPARATUS USED	1. Operating Systems (64-Bit)64-BIT Fedora 17 or latest 64-BIT Update of Equivalent Open source OS or latest 64-BIT Version 2. Programming Tools (64-Bit): Android Studio v 2.3.3 3. Java Development Kit 8
REFERENCES	• Neil Smyth, “Android Studio 2 Development Essentials”, Payload Media, ISBN: 1532853319 • John Horton, “Android Programming for Beginners”, ISBN 10:1785883267 • Reto Meier, “Professional Android 4 Application Development”, Wrox, ISBN-10: 1118102274; ISBN-13: 978-1118102275 • Greg Nudelman, “Android Design Patterns :Interaction Design Solutions for Developers”, ISBN-10: 1118394151; ISBN-13: 978-1118394151
INSTRUCTIONS FOR WRITING JOURNAL	**1. Date** **2. Assignment no.** **3. Problem definition** **4. Learning objective** **5. Learning Outcome** **6. State Transition Diagram** **7. Concepts related Theory** **8. Program code with proper documentation.** **9. Output of program.** **10. Conclusion and applications (the verification and testing of outcomes).**

- **Aim**
  Design a mobile app for media player to load data from internal or external storage.

- **Prerequisites**
  - ■ Object oriented programming concepts
  - ■ Programming features in java
  - ■ Installation of android studio

- **Learning Objectives**
  To understand design and development in Android studio
  To learn memory management in android

- **Learning Outcome**
  After successfully completing this assignment, you should be able to

- Design and implement music player application on android studio.

- **Concepts related Theory**
  Music player or media player is an application to play media like songs and videos on the device it is intended to be used. Music player in android requires layout to be designed, songs to be listed and played providing information about the song played and provisions of Fast forward, Fast backward, play and pause, next and previous and also a seek bar to give the information on time period to end users.

- Building the music player will involve using the *ContentResolver* class to retrieve tracks on the device, the *MediaPlayer* class to play audio and the *MediaController* class to control playback. We will also use a *Service* instance to play audio when the user is not directly interacting with the app.

- Following are the simple steps to build the music player application in android:

  - ■ Create and Configure a New Project
  - ■ Query the Device for Songs
  - ■ Display the songs

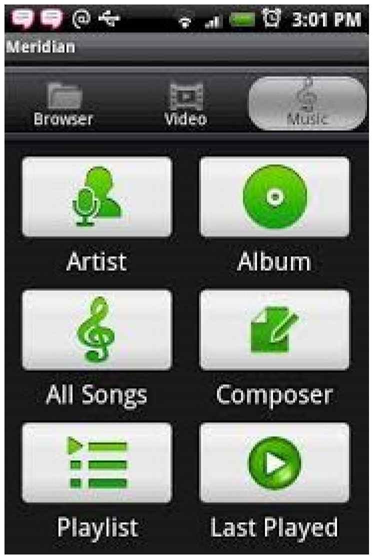


Fig: Screenshots of the music player

- **Test cases:**
  Positive Testing:

Test case ID	Test Case Name	Test Case Description	Expected Result	Actual Result	Status
MP1	Launch app	Launch the application	must open successfully	Opened successfully without errors	Pass
MP2	List the songs	List all the songs with mp3 format from internal and external storage of the device	All the songs must be listed	List of the songs is displayed	Pass
MP3	Play songs	Mp3 must play and audible on clicking play button or selecting a song from the list	Song must play	Song played	Pass
MP4	Audio controls	Next, previous, fast forward and backward, seek bar controls should work properly	All the audio controls must be working properly	Controls properly working	Pass

MP5	Refresh the list	List should be refreshed after a click on refresh button	Refreshed list must be displayed	Refreshed list is displayed	Pass

<u>Negative Testing:</u>

Test case ID	Test Case Name	Test Case Description	Expected Result	Actual Result	Status
MP6	Invalid formats	Files of other formats such as mp4, 3gp,doc,odt, etc should be ignored in the list	Only mp3 files must be listed and other files are excluded.	Only mo3 files are displayed	Pass
MP7	Out of bound exception	External Storage must be limited considering mobile application. Max of 64GB of external storage is supported.	Refresh button should work properly for external storage of 64GB MicroSD card.	Works properly for 32GB MicroSD card and hangs on 64GB card.	Moderately pass

Conclusion:

Music player application in android is implemented and tested successfully.

Oral Questions on Android:

1) What is a visible activity?

A visible activity is one that sits behind a foreground dialog. It is actually visible to the user, but not necessarily being in the foreground itself.

2) When is the best time to kill a foreground activity?

The foreground activity, being the most important among the other states, is only killed or terminated as a last resort, especially if it is already consuming too much memory. When a memory paging state has been reach by a foreground activity, then it is killed so that the user interface can retain its responsiveness to the user.

3) Is it possible to use or add a fragment without using a user interface?

Yes, it is possible to do that, such as when you want to create a background behavior for a particular activity. You can do this by using add(Fragment,string) method to add a fragment from the activity.

4) How do you remove icons and widgets from the main screen of the Android device?

To remove an icon or shortcut, press and hold that icon. You then drag it downwards to the lower part of the screen where a remove button appears.

5) What are the core components under the Android application architecture?

There are 5 key components under the Android application architecture:
– services
– intent
– resource externalization
– notifications
– content providers

6) What composes a typical Android application project?

A project under Android development, upon compilation, becomes an .apk file. This apk file format is actually made up of the AndroidManifest.xml file, application code, resource files, and other related files.

7) What is a Sticky Intent?

A Sticky Intent is a broadcast from sendStickyBroadcast() method such that the intent floats around even after the broadcast, allowing others to collect data from it.

8) Do all mobile phones support the latest Android operating system?

Some Android-powered phone allows you to upgrade to the higher Android operating system version. However, not all upgrades would allow you to get the latest version. It depends largely on the capability and specs of the phone, whether it can support the newer features available under the latest Android version.

9) What is portable wi-fi hotspot?

Portable Wi-Fi Hotspot allows you to share your mobile internet connection to other wireless device. For example, using your Android-powered phone as a Wi-Fi Hotspot, you can use your laptop to connect to the Internet using that access point.

10) What is an action?

In Android development, an action is what the intent sender wants to do or expected to get as a response. Most application functionality is based on the intended action.

11) What is the difference between a regular bitmap and a nine-patch image?

In general, a Nine-patch image allows resizing that can be used as background or other image size requirements for the target device. The Nine-patch refers to the way you can resize the image: 4 corners that are unscaled, 4 edges that are scaled in 1 axis, and the middle one that can be scaled into both axes.

12) What language is supported by Android for application development?

The main language supported is Java programming language. Java is the most popular language for app development, which makes it ideal even for new Android developers to quickly learn to create and deploy applications in the Android environment.

13) What is the proper way of setting up an Android-powered device for app development?

The following are steps to be followed prior to actual application development in an Android-powered device:
-Declare your application as “debuggable” in your Android Manifest.
-Turn on “USB Debugging” on your device.
-Set up your system to detect your device.

14) Enumerate the steps in creating a bounded service through AIDL.

1. create the .aidl file, which defines the programming interface
2. implement the interface, which involves extending the inner abstract Stub class as well as implanting its methods.
3. expose the interface, which involves implementing the service to the clients.

15) What is the importance of Default Resources?

When default resources, which contain default strings and files, are not present, an error will occur and the app will not run. Resources are placed in specially named subdirectories under the project res/ directory.

16) When dealing with multiple resources, which one takes precedence?

Assuming that all of these multiple resources are able to match the configuration of a device, the ‘locale’ qualifier almost always takes the highest precedence over the others.

Revised on: 15-06-2019

ASSINGMENT NO.	**7**
TITLE	**Design a mobile app using Google Map and GPS to trace the location.**
PROBLEM STATEMENT / DEFINITION	Design a mobile app using Google Map and GPS to trace the location.
OBJECTIVE	To implement mobile app to trace the location
S/W PACKAGES AND HARDWARE APPARATUS USED	1.Operating Systems (64-Bit)64-BIT Fedora 20 or latest 64-BIT Update of Equivalent Open source OS 2. Latest Android studio
REFERENCES	• Neil Smyth, ―Android Studio 2 Development Essentials‖, Payload Media, ISBN: 1532853319 • John Horton, ―Android Programming for Beginners‖, ISBN 10: 1785883267 ISBN 13:9781785883262 • Reto Meier, ―Professional Android 4 Application Development‖, Wrox, ISBN-10:1118102274; ISBN-13: 978-1118102275
INSTRUCTIONS FOR WRITING JOURNAL	• **Date** • **Assignment no.** • **Problem definition** • **Learning objective** • **Learning Outcome** • **Concepts related Theory** • **Program code with proper documentation.** • **Output of program.** • **Conclusion and applications (the verification and testing of outcomes).**

- **Aim**

  Design a mobile app using Google Map and GPS to trace the location.

- **Prerequisites**

  - Concept of various attributes such as
    1. Information about Android Studio
    2. Information about Google Map
    3. Information about GPS
  - Object oriented programming features.

- **Learning Objectives**

  - To understand the basic android studio.
  - To understand the basic Google map and GPS.

- **Learning Outcome**

After successfully completing this assignment, Student should be able to Understand & Implement mobile app to trace the location.

- **Concepts related Theory**

Google Maps is a Web-based service that provides detailed information about geographical regions and sites around the world. In addition to conventional road maps, Google Maps offers aerial and satellite views of many places. In some cities, Google Maps offers street views comprising photographs taken from vehicles.

With Google Maps installed on your device, you can view street and satellite maps of the whole World. Not only this, but it can be used to plot routes, find local places of interest, socialize with people around you and 'walk' along roads with Google's Street View.

Google Maps is incredibly easy to use on your Android device. The application automatically detects your current location and displays it on screen. You can **move around by holding your finger** and dragging the screen and zoom in and out by pinching with your fingers.

The app allows you to **save maps offline** and manage them from an easy-to-access list. The app also shows you the total walking time of your trip and when the next bus or train is. **Turn-by-turn navigation** shows you distance and estimated arrival time, and gives you access to alternative routes and features lane assistance.

Maps also includes a new Explore feature , which shows you different places and activities around you. You can filter by distance, time of day, and type of place and get business information. So you won't be seeing results of restaurants that are closed, as Google Maps will detect the time of day. It is even aware of weather conditions, making sure you don't head to a park when it's about to rain. You can also search for places you've reviewed or saved using the "Your Places" option.

Google Map Layout File:

<Fragment

android:id="@+id/maps"

android:layout_width="matchparent"

android:layout_height="matchparent"

Google Map Android Manifest File:

We have to add the permission along with the google map API key in Android Manifest.

Permission:

1) ACCESS_FINE_LOCATION -> GPS location

2) ACCESS_COARSE_LOCATION -> permission for network provider location

Syntax:

< user_permission android:name="android permission:permission_type"/>

<!---------Google API Key----------->

<metadata

android:name="package_path"

android:value="Google API Key"/>

Customizing Google Maps:

1) Adding marker -> Done using addMarker() in Google Map android Manifest

addmarker(Googlemap_Android Manifest().position(0)title("Myloc"));

2) changing Map Type ->

Syntax -> Googlemap.SetMapType((Googlemap_MAPTYPE));

3) Enable/Disable Zoom ->

googlemap.getvisitthings().SetZoomGesturesEnabled(true);

4) To Get Current Location -> getmyLocation()

5) To Zoom a Particular Area -> map,moveCamera(CameraUpdate,up);

Refer the below given link for additional information

https://github.com/googlemaps/

Conclusion:

Thus, after successfully completing this assignment, Students should be able to understand & Implement Mobile app to trace the location.