ADVANCED OPERATING SYSTEMS

INDEX

	The address 19986 contains: page number = 4 offset = 3602.	

6.	Write a Java program that simulates the following disk-scheduling algorithms. Design separate classes that implement the following scheduling algorithms:
	a. FCFS
	b. SSTF
	c. SCAN
	d. C-SCAN
	e. LOOK
	Each algorithm will implement the following interface:
	public interface DiskScheduler
	{
	// service the requests
	// return the amount of head movement
	// for the particular algorithm
	public int serviceRequests();
	}
	The serviceRequests() method will return the amount of head movement required by the disk-scheduling algorithm.	

7.	Write a program that implements the FIFO and LRU page-replacement algorithms presented in this chapter. First, generate a random page reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithms so that the number of page frames can vary as well. Assume that demand paging is used. Design and implement two classes—LRU and FIFO—that extend ReplacementAlgorithm. Each of these classes will implement the insert() method, one class using the LRU page-replacement algorithm and the other using the FIFO algorithm. Test your algorithm with suitable Java programs.	
8.	Using Worker thread write Android code for a click listener that downloads an image from a separate thread and displays it in an ImageView.	
9.	Write Android activity that includes each of the fundamental lifecycle methods	
10	Write Android application to demonstrate data storage with following options (any one can be asked in Practical examination): Shared Preferences (Store private primitive data in key-value pairs) Internal Storage (Store private data on the device memory) External Storage (Store public data on the shared	

	external storage) SQLite Databases (Store structured data in a private database) Network Connection (Store data on the web with your own network server).	

Practical 1

Port 17 is known as the 'Quote of the day service'. When a client connects to port 17 on a server, the server responds with a quote for that day. Write a server program so that it delivers a quote of the day. The quotes should be printable ASCII characters and should contain fewer than 512 characters, although multiple lines are allowed. Since port 17 is considered well known and therefore unavailable, have your server listen to port 6017. Write the client code used to read the quotes returned by the server.

Code:

P1Server.java

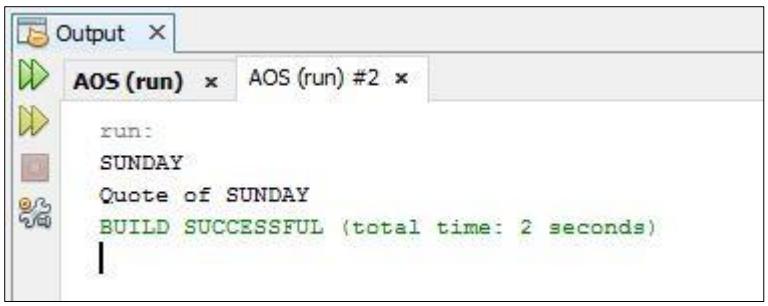
```
import java.io.*;
import java.net.*;
import java.util.*;

public class P1Server {
   public static void main(String ar[]) {
      try {
         String day = null;
         String quote = null;
         Calendar c = Calendar.getInstance();
         Date dt = new Date();
         c.setTime(dt);
         int dow = c.get(Calendar.DAY_OF_WEEK);
         switch (dow) {
            case 1: day = "SUNDAY"; quote = "Quote of SUNDAY";
           break;
            case 2: day = "MONDAY"; quote = "Quote of MONDAY";
           break;
            case 3: day = "TUESDAY"; quote = "Quote of TUESDAY";
           break;
            case 4: day = "WEDNESDAY";quote = "Quote of WEDNESDAY";
           `break;
            case 5: day = "THURSDAY"; quote = "Quote of THURSDAY";
           break;
            case 6: day = "FRIDAY"; quote = "Quote of FRIDAY";
           break;
            case 7: day = "SATURDAY"; quote = "Quote of SATURDAY";
           break;
                    }
         ServerSocket sk = new ServerSocket(6017);
         while (true) {
            Socket cl = sk.accept();
            PrintWriter p = new PrintWriter(cl.getOutputStream(), true);
            if (quote.length() <= 512) {
               p.println(day + "\n" + quote);
            }
           else {
               p.println("more than 512 char");}
```

```
            cl.close();   }
        } catch (IOException e) {
            System.out.println(e);}}}
```

P1Client.java

```
import java.io.*;
import java.net.*;
public class P1Client {
    public static void main(String ar[]) {
        InputStream i = null;
        BufferedReader b = null;
        Socket s = null;
        try {  s = new Socket(InetAddress.getLocalHost(), 6017);
            i = s.getInputStream();
            b = new BufferedReader(new InputStreamReader(i));
            String l;
            while ((l = b.readLine()) != null) {
                System.out.println(l); }
        } catch (IOException e) {
            System.out.println(e);
        } finally {
            try {s.close();
            } catch (Exception e) {
                System.out.println(e.getMessage());}}}}
```

Output:



Practical 2

Write a client–server application using Java sockets that allows a client to write a message (as a String) to a socket. A server will read this message, count the number of characters and digits in the message, and send these two counts back to the client. The server will listen to port 6100. The client can obtain the String message that it is to pass to the server either from the command line or by using a prompt to the user. One strategy for sending the two counts back to the client is for the server to construct an object containing:
a. The message it receives from the client
b. A count of the number of characters in the message
c. A count of the number of digits in the message.

Code:

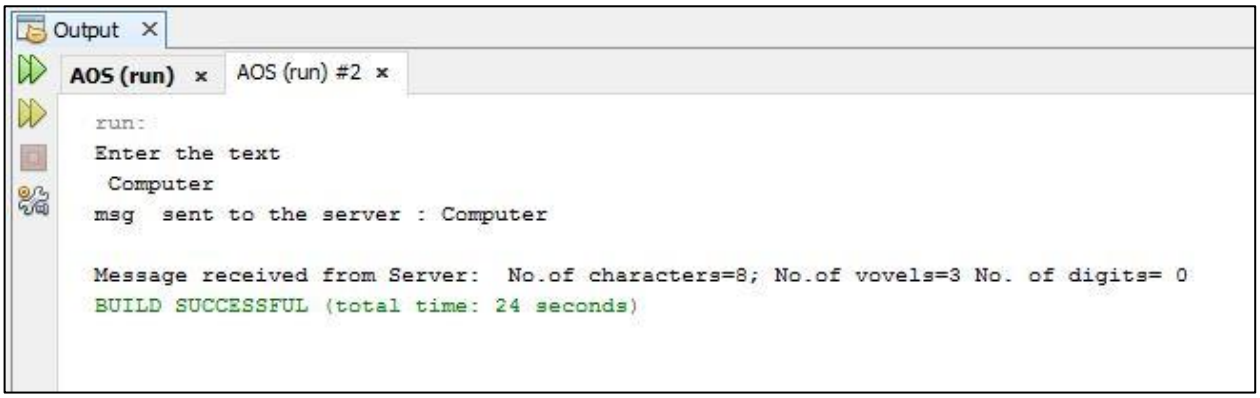
P2Server.java

```
import java.io.*;
import java.net.*;
public class P2Server {
   private static Socket socket;
   public static void main(String args[]) {
      try {
         ServerSocket serverSocket = new ServerSocket(6017);
         System.out.println("Server started and listening to the port 6102");
         while (true) {
            socket = serverSocket.accept();
            InputStream is = socket.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            String msg = br.readLine();
            System.out.println("Message received from client is " + msg);
            String returnMessage = "";
            try {
               int d = 0;  int v = 0;
               for (int i = 0; i < msg.length(); ++i) {
                  char k = msg.charAt(i);
                  if (Character.isDigit(k)) {
                     d++; }
                  if (k == 'a' || k == 'e' || k == 'i' || k == 'o' || k == 'u') {
                     v++;}}
               returnMessage = " No.of characters=" + msg.length() + "; No.of vovels=" + v + " No. of digits= " + d +
"\n";
            } catch (Exception e) {}
            OutputStreamWriter osw = new OutputStreamWriter(socket.getOutputStream());
            BufferedWriter bw = new BufferedWriter(osw);
            bw.write(returnMessage);
            System.out.println("Message sent to the client is " + returnMessage);
            bw.flush();}}catch (Exception e) {
         e.printStackTrace();} finally {
         try {socket.close();
         } catch (Exception e) {}}}}
```

P2Client.java

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
public class P2Client {
   private static Socket s;
   public static void main(String[] args) throws IOException {
      Scanner sc = new Scanner(System.in);
      try{
            s=new Socket("127.0.0.1",6017);
           OutputStream o=s.getOutputStream();
             OutputStreamWriter os=new  OutputStreamWriter(o);
             BufferedWriter b=new BufferedWriter(os);
             System.out.println("Enter the text");
             String m=(sc.next())+"\n";
              b.write(m);
             b.flush();
             System.out.println("msg  sent to the server : "+ m);
             InputStream is=s.getInputStream();
             InputStreamReader ir=new InputStreamReader(is);
             BufferedReader br=new BufferedReader(ir);
              String mg= br.readLine();
              System.out.println("Message received from Server: "+mg);
      }
      catch(Exception e) {
           e.printStackTrace();
      }
      finally{
         try{ s.close();}
catch(Exception e) {
              e.printStackTrace();}  }}}
```

Output:



Practical 3

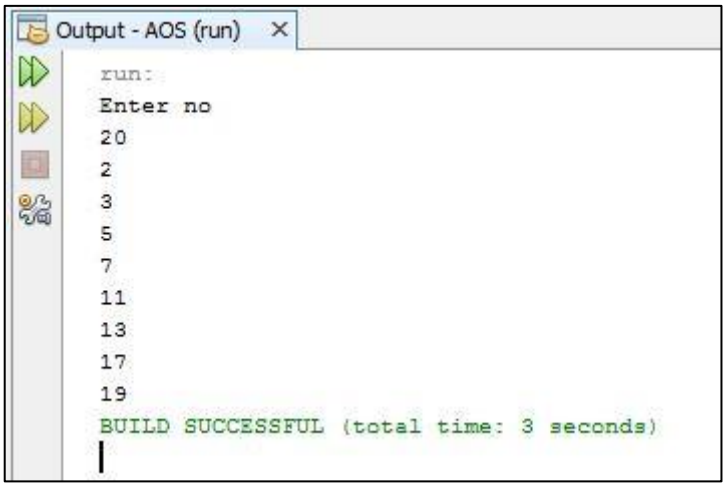
Write a multithreaded Java program that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.

Code:

P3.java

```
import java.util.Scanner;
public class Prime_number extends Thread {
static int x;
  void prime(int n){
    for (int i = 2; i < n; i++) {
      int flag = 0;
      for (int j = 2; j < i; j++) {
        if (i%j==0) {
          flag=1;
        }}if(flag==0){
          System.out.println(i);}}}
  public void run(){
    prime(x);
  }
  public static void main(String[] args) {
    Prime_number pm = new Prime_number();
    System.out.println("Enter no");
    Scanner sc = new Scanner(System.in);
    x = sc.nextInt();
    pm.start();}}
```

Output:



Practical 4

Servers can be designed to limit the number of open connections. For example, a server may wish to have only N socket connections open at any point in time. After N connections have been made, the server will not accept another incoming connection until an existing connection is released. Write Java programs to demonstrate the scenario.

Code:

P4Server.java

```
import java.io.IOException;
import java.net.*;
public class P4Server {
public static int counter = 0;
public static void main(String[] args) {
    try {
      ServerSocket server;
      server = new ServerSocket(4444);
      System.out.println("Running.. .");
      while (true) {
        Server pm = new Server();
Socket socket = server.accept();
        pm.counter++;
        new Thread(new Operations(socket, pm.counter)).start();}
    } catch (IOException ex) {
      System.out.println(ex);}}}
```

P4Operation.java

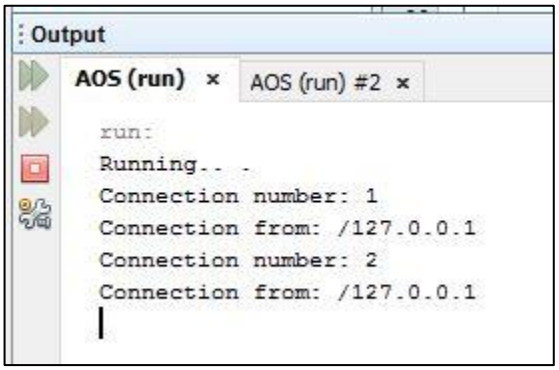
```
import java.io.*;
import java.net.Socket;
public class P4Operation implements Runnable {
  Socket socket;
  PrintWriter out;
  BufferedReader in;
  int counter = 0;
  Server pm = new Server();
  public Operations(Socket socket, int counter) {
    this.socket = socket;
    this.counter = counter;
    System.out.println("Connection number: " + this.counter);
  }
  public void run() {
    try {
      in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
      out = new PrintWriter(socket.getOutputStream(), true);
      if (this.counter > 3) {
        out.println("505");
        System.out.println("Request from " + socket.getInetAddress() + " declined");
      } else {
        System.out.println("Connection from: " + socket.getInetAddress());
        out.println("You are Client No " + counter);}}
      catch (IOException | NumberFormatException e) {
```

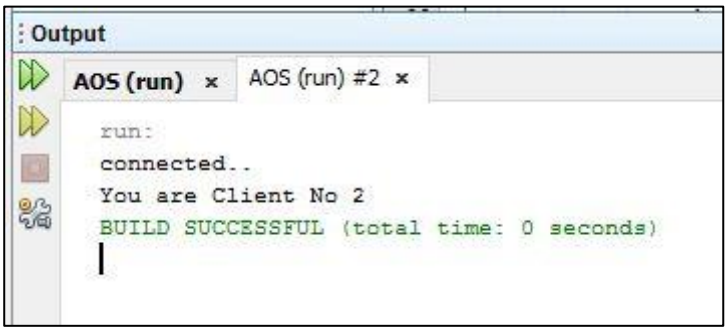
```
        System.out.println(e);}}}
```

P4Client.java

```
import java.io.*;
import java.net.Socket;
import java.util.Scanner;
public class Client {
public static void main(String[] args) {
    int range;
    PrintWriter out;
    BufferedReader incoming;
    Socket socket = null;
    Scanner in = new Scanner(System.in);
try {
        socket = new Socket("127.0.0.1", 4444);
        System.out.println("connected..");
        out = new PrintWriter(socket.getOutputStream(), true);
        incoming = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        String result = incoming.readLine();
        if (result.equals("505")) {
           System.out.println("Connection refused due to too much client");
           socket.close();
        } else {
           System.out.println(result);}}
       catch (Exception e) {
        System.out.println(e);}}}
```

Output:





Practical 5:

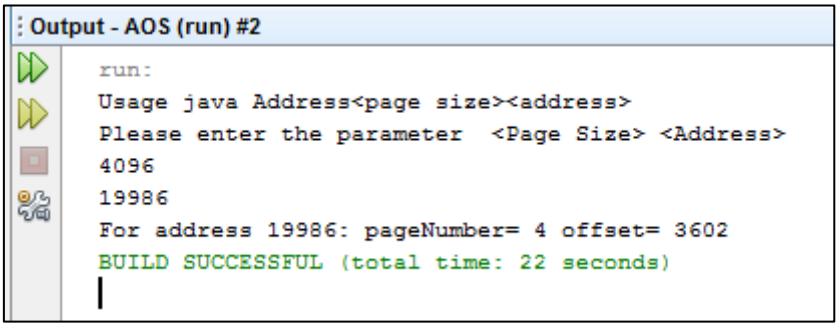
Assuming that a system has a 32-bit virtual address, write a Java program that is passed
(1) the size of a page and
(2) the virtual address.
Your program will report the page number and offset of the given virtual address with the specified page size. Page sizes must be specified as a power of 2 and within the range 1024 — 16384 (inclusive). Assuming such a program is named Address, it would run as follows:
java Address 4096 19986
and the correct output would appear as:
The address 19986 contains:
page number = 4
offset = 3602.

Code:
P5.java

```
import java.io.*;
public class P5 {
   public static final int ADDRESS_SIZE = 32;
   public static void main(String args[]) {
      try {
         if (args.length != 2) {
            System.out.println("Usage java Address<page size><address>");}
         System.out.println("Please enter the parameter  <Page Size> <Address>");
         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
         int ps = Integer.parseInt(br.readLine().trim());
         int ad = Integer.parseInt(br.readLine().trim());
         int pageBits = 0;
         int pageMask = 0;
         int offSetMask = 0;
         switch (ps) {
            case 1024:
               pageBits = 10;
               offSetMask = 0x000003ff;
               pageMask = 0xfffffc00;
               break;
            case 2048:
               pageBits = 11;
               offSetMask = 0x000007fd;
               pageMask = 0xfffffa72;
               break;
            case 4096:
               pageBits = 12;
               offSetMask = 0x00000fff;
               pageMask = 0xfffff000;
               break;
            case 8192:
               pageBits = 13;
               offSetMask = 0x00001fff;
               pageMask = 0xffffe000;
               break;
```

```
        case 16384:
            pageBits = 14;
            offSetMask = 0x000003ff;
            pageMask = 0xfffffc00;
        break;
        default: System.out.println("give proper address"); }
    int pn = (ad & pageMask) >> pageBits;
    int os = (ad & offSetMask);
    System.out.println("For address " + ad + ": pageNumber= " + pn + " offset= " + os);
  } catch (Exception e) {e.printStackTrace();}}}
```

Output:



Practical 6

Write a Java program that simulates the following disk-scheduling algorithms. Design separate classes that implement the following scheduling algorithms:

a. FCFS
b. SSTF
c. SCAN
d. C-SCAN
e. LOOK

Each algorithm will implement the following interface:

```
public interface DiskScheduler
{
// service the requests
// return the amount of head movement
// for the particular algorithm
public int serviceRequests();
}
```

The serviceRequests() method will return the amount of head movement required by the disk-scheduling algorithm.

Code:

DiskScheduler.java

```
public interface DiskScheduler
{
	// service the requests
	// return the amount of head movement
	// for the particular algorithm
	public int serviceRequests();
}
```

Generator.java

```
public class Generator{
	private static final int DEFAULT_SIZE = 100;
	private static final int RANGE = 99;
	int[] referenceString;
	public Generator() {
		this(DEFAULT_SIZE); }
	public Generator(int count) {
		if (count < 0)
			throw new IllegalArgumentException();
		java.util.Random generator = new java.util.Random();
		referenceString = new int[count];
		for (int i = 0; i < count; i++)
			referenceString[i] = generator.nextInt(RANGE + 1); }
	public int[] getCylinders() {
		return referenceString;}}
```

FCFS.java

```
class FCFS implements DiskScheduler{
	private static int[] rString;
	private static int start;
```

```
        private static int sum;
public FCFS(int[] rS, int s){
                this.rString = new int[rS.length];
                System.arraycopy(rS,0,rString,0,rS.length);
                this.start = s;
                this.sum=0; }
        public int serviceRequests(){
                sum += Math.abs(rString[0] - start);
                for(int i=1;i<rString.length;i++){
                        //sum += Math.min(Math.abs(rString[i]-rString[i-1]),99-Math.abs(rString[i]-rString[i-
1]));
                        sum += Math.abs(rString[i]-rString[i-1]);}
                return sum;}
        public static void main(String[] args){
    //Generator ref = new Generator();
                //int[] referenceString = ref.getCylinders();
                int[] referenceString = {98,183,37,122,14,124,65,67};
                DiskScheduler fcfs = new FCFS(referenceString, 53);
                System.out.println("FCFS = " + fcfs.serviceRequests());}}
```

SSTF.java

```
import java.util.ArrayList;
class SSTF implements DiskScheduler{
        private static int[] rString;
        private static int start;
        private static int sum;
        public SSTF(int[] rS, int s){
                this.rString = new int[rS.length];
                System.arraycopy(rS,0,rString,0,rS.length);
                this.start = s; this.sum=0;}
        public int serviceRequests(){
    int pos;
    int currentPos = this.start;
    int sum = 0;
    ArrayList<Integer> list = new ArrayList<Integer>();
    for(int i=0;i<this.rString.length; i++){
      list.add(new Integer(this.rString[i]));
    }
    for(int i=0; i<this.rString.length; i++){
      pos = this.findNearestPos(list, currentPos);
      sum += Math.abs(currentPos-pos); currentPos = pos;}
    return sum;}
  private int findNearestPos(ArrayList<Integer> list, int currentPos){
    int minDistance = Math.abs(currentPos-list.get(0));
    int distance; int pos = 0;
    int counter = 0;
    for(int i : list){
      distance = Math.abs(currentPos-i);
      if(distance<minDistance){
        minDistance = distance;
        pos = counter;}
      counter++;
```

```
    }
    int ret = list.get(pos);
    list.remove(pos);
    return ret;]  }
        public static void main(String[] args){
    Generator ref = new Generator();
        int[] referenceString = ref.getCylinders();
                DiskScheduler sstf = new SSTF(referenceString, 13);
                System.out.println("SSTF = " + sstf.serviceRequests());}}
```

Scan.java

```
import java.util.ArrayList;
class Scan implements DiskScheduler{
        private static int[] rString;
        private static int start;
        private static int sum;
  private static int RANGE = 100;
        public Scan(int[] rS, int s){
                this.rString = new int[rS.length];
                System.arraycopy(rS,0,rString,0,rS.length);
                this.start = s;
                this.sum=0;}
  public int serviceRequests(){
    ArrayList<Integer> list = new ArrayList<Integer>();
    for(int i=0; i<this.rString.length; i++){
      list.add(new Integer(this.rString[i]));
    }
    int direction = -1;
    int currentPos = this.start;
    int sum = 0;
    while(!list.isEmpty()){
      currentPos += direction;
      sum++;
      if(list.contains(new Integer(currentPos))){
        list.remove(new Integer(currentPos)); }
      if(currentPos == 0 || currentPos == this.RANGE){
        direction = -direction;} }
    return sum;   }
        public static void main(String[] args){
    //Generator ref = new Generator();
                //int[] referenceString = ref.getCylinders();
                int[] referenceString = {98,183,37,122,14,124,65,67};
                DiskScheduler scan = new Scan(referenceString, 53);
        System.out.println("SCAN = " + scan.serviceRequests());}}
```

CScan.java

```
import java.util.ArrayList;
class CScan implements DiskScheduler{
        private static int[] rString;
        private static int start;
        private static int sum;
```

```
  private static int RANGE = 100;
        public CScan(int[] rS, int s){
                this.rString = new int[rS.length];
                System.arraycopy(rS,0,rString,0,rS.length);
                this.start = s;
                this.sum=0;}
  public int serviceRequests(){
    ArrayList<Integer> list = new ArrayList<Integer>();
    for(int i=0; i<this.rString.length; i++){
      list.add(new Integer(this.rString[i]));
    }
    int direction = 1;
    int currentPos = this.start;
    int sum = 0;
    while(!list.isEmpty()){
      currentPos += direction;
      sum++;
      if(list.contains(new Integer(currentPos))){
        list.remove(new Integer(currentPos));
      }
      if(currentPos == this.RANGE){
        sum += this.RANGE;
        currentPos = -1;}}
    return sum;}
        public static void main(String[] args){
    //Generator ref = new Generator();
        //int[] referenceString = ref.getCylinders();
                int[] referenceString = {98,183,37,122,14,124,65,67};
                DiskScheduler cscan = new CScan(referenceString, 53);
                System.out.println("CSCAN = " + cscan.serviceRequests());}}
```

Look.java

```
import java.util.ArrayList;
class Look implements DiskScheduler{
        private static int[] rString;
        private static int start;
        private static int sum;
        public Look(int[] rS, int s){
                this.rString = new int[rS.length];
                System.arraycopy(rS,0,rString,0,rS.length);
                this.start = s;
                this.sum=0;}
  public int min(int[] array){
    int min = array[0];
    for(int i: array){
      if(i<min){
        min = i;}}
    return min;}
  public int max(int[] array){
    int max = array[0];
    for(int i: array){
```

```
        if(i>max){
            max = i;}}
    return max;}
  public int serviceRequests(){
    int max = this.max(this.rString);
    int min = this.min(this.rString);
    ArrayList<Integer> list = new ArrayList<Integer>();
    for(int i=0; i<this.rString.length; i++){
        list.add(new Integer(this.rString[i]));
    }
    int direction = -1;
    int currentPos = this.start;
    int sum = 0;
    while(!list.isEmpty()){
        currentPos += direction;
        sum++;
        if(list.contains(new Integer(currentPos))){
            list.remove(new Integer(currentPos));
        }
        if(currentPos == min || currentPos == max){
            direction = -direction;}}
    return sum;
  }
        public static void main(String[] args){
    //Generator ref = new Generator();
                //int[] referenceString = ref.getCylinders();
                int[] referenceString = {98,183,37,122,14,124,65,67};
                DiskScheduler look = new Look(referenceString, 53);
                System.out.println("LOOK = " + look.serviceRequests());}}
```

P6.java

```
class P6{
        public static void main(String[] args){
    Generator ref = new Generator();
                int[] referenceString = ref.getCylinders();
            System.out.println("Amount of head movement for");
                //int[] referenceString = {98,183,37,122,14,124,65,67};
                DiskScheduler fcfs = new FCFS(referenceString, 53);
                System.out.println("FCFS = " + fcfs.serviceRequests());
    DiskScheduler sstf = new SSTF(referenceString, 53);
                System.out.println("SSTF = " + sstf.serviceRequests());
    DiskScheduler scan = new Scan(referenceString, 53);
                System.out.println("SCAN = " + scan.serviceRequests());
    DiskScheduler cscan = new CScan(referenceString, 53);
                System.out.println("CSCAN = " + cscan.serviceRequests());
    DiskScheduler look = new Look(referenceString, 53);
                System.out.println("LOOK = " + look.serviceRequests());
  }
}
```

Output:

```
Output - OS (run) #2
run:
Amount of head movement for
FCFS = 3179
SSTF = 143
SCAN = 335
CSCAN = 753
LOOK = 347
BUILD SUCCESSFUL (total time: 2 seconds)
```

Practical 7

Write a program that implements the FIFO and LRU page-replacement algorithms presented in this chapter. First, generate a random page reference string where page numbers range from 0 to 9. Apply the random page-reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithms so that the number of page frames can vary as well. Assume that demand paging is used. Design and implement two classes—LRU and FIFO—that extend ReplacementAlgorithm. Each of these classes will implement the insert() method, one class using the LRU page-replacement algorithm and the other using the FIFO algorithm. Test your algorithm with suitable Java programs.

Code:

PageGenerator.java

```
public class PageGenerator
{
	private static final int DEFAULT_SIZE = 100;
	private static final int RANGE = 9;
	int[] referenceString;
	public PageGenerator() {
		this(DEFAULT_SIZE);   }
	public PageGenerator(int count) {
		if (count < 0)
			throw new IllegalArgumentException();
		java.util.Random generator = new java.util.Random();
		referenceString = new int[count];
		for (int i = 0; i < count; i++)
			referenceString[i] = generator.nextInt(RANGE + 1);    }
	public int[] getReferenceString() {
		return referenceString;}}
```

ReplacementAlgorithm.java

```
public abstract class ReplacementAlgorithm{
	// the number of page faults
	protected int pageFaultCount;
	// the number of physical page frame
	protected int pageFrameCount;
	public ReplacementAlgorithm(int pageFrameCount) {
		if (pageFrameCount < 0)
			throw new IllegalArgumentException();
			this.pageFrameCount = pageFrameCount;
		pageFaultCount = 0;
	}
	public int getPageFaultCount() {
	return pageFaultCount;}
	public abstract void insert(int pageNumber); }
```

FIFO.java

```
public class FIFO extends ReplacementAlgorithm {
	private int[] table;
```

```
private int nextPg; // The next page to be replaced, if a new insert requires it
private int numFaults;
// Constructor
public FIFO(int pageFrameCount) {
	super(pageFrameCount);
	table = new int[pageFrameCount];
	for (int i = 0; i < table.length; i++) {
		table[i] = -1;}
	nextPg = 0;
	numFaults = 0; }
// Returns the number of page faults that have occured so far
public int getPageFaultCount() {
	return numFaults;}
// Returns true only if the given page number is currently in the page table
public boolean isIn(int pageNumber) {
	boolean bool = false;
	for (int i = 0; i < table.length; i++) {
		if (table[i] == pageNumber) bool = true;}
	return bool;}
// Insert a new page using the FIFO algorithm
@Override
public void insert(int pageNumber) {
	// Check if there's a page fault
	boolean pf = !isIn(pageNumber);
	if (pf) {
		// Insert the page into the table and up the page fault count
		table[nextPg] = pageNumber;
		numFaults++;   }
	// Print out the contents of the page table
	for (int i = 0; i < pageFrameCount; i++) {
		System.out.print(i+1+" - ");
		if (table[i] == -1) {
			System.out.print("(empty)");
		} else {
			System.out.print(table[i]);
			if (pf && nextPg == i) {
				System.out.print(" (pf)");}}
		System.out.print("\n");}
	System.out.print("\n");
		if (pf) {
		nextPg++; // Get ready for the next page-faulting insert
		if (nextPg > table.length-1) { // Keep it in bounds
			nextPg = 0;}}}}
```

LRU.java

```
public class LRU extends ReplacementAlgorithm {
	private int[] table;
	private int nextPg; // The next page to be replaced, if a new insert requires it
	private int numFaults;
	private int[] clocks; // Sequence of pages
	private int clock; // Logical clock ticks
```

```
// Constructor
public LRU(int pageFrameCount, int[] refStr) {
	super(pageFrameCount);
	table = new int[pageFrameCount];
	for (int i = 0; i < table.length; i++) {
		table[i] = -1; }
	nextPg = 0;
	numFaults = 0;
	clocks = new int[pageFrameCount];
	clock = 0; }
// Returns the number of page faults that have occured so far
public int getPageFaultCount() {
	return numFaults;}
// Returns true only if the given page number is currently in the page table
public boolean isIn(int pageNumber) {
	boolean bool = false;
	for (int i = 0; i < table.length; i++) {
		if (table[i] == pageNumber) bool = true;}
	return bool;}
// Determines which page to replace by seeing which was used the farthest in the past
private void setNextPg() {
	int curHighest = 0;
	// See if any page frames are empty (available)
	for (int i = 0; i < table.length; i++) {
		if (table[i] == -1) {
			curHighest = i;
			nextPg = curHighest;
			return;}}
	for (int i = 1; i < clocks.length; i++) {
		if (clocks[i] < clocks[curHighest]) {
			curHighest = i;}}
nextPg = curHighest;	}
// Insert a new page using the OPT algorithm
@Override
public void insert(int pageNumber) {
	clock++; // Update the clock
	// Check if there's a page fault
	boolean pf = !isIn(pageNumber);
		if (pf) {
		// Insert the page into the table and up the page fault count
		setNextPg();
		table[nextPg] = pageNumber;
		clocks[nextPg] = clock;
		numFaults++;
	} else { // If it's already in the page table, update the clock
		for (int i = 0; i < table.length; i++) {
			if (table[i] == pageNumber) {
				clocks[i] = clock;}}}
	// Print out the contents of the page table
	for (int i = 0; i < pageFrameCount; i++) {
		System.out.print(i+1+" - ");
```

```
                if (table[i] == -1) {
                        System.out.print("(empty)");
                } else {  System.out.print(table[i]);
                        if (pf && nextPg == i) {
                                System.out.print(" (pf)");}}
                System.out.print("\n");}
        System.out.print("\n");}}
```

Test.java

```
public class Test{
        public static void main(String[] args) {
                // Check command-line input
                if (args.length != 2 ) {
                        System.out.println("Must provide 2 numbers on the command-line: the reference string
size, and the number of page frames!");
                        return; }
                PageGenerator ref = new PageGenerator(new Integer(args[0]).intValue());
                int[] referenceString = ref.getReferenceString();
                // Print out the reference string
                System.out.print("Reference string = ");
                for (int i = 0; i < referenceString.length; i++) {
                        System.out.print(referenceString[i]+" ");}
                System.out.println("\n");
                // FIFO Algorithm
                System.out.println("FIFO Algorithm\n");
                ReplacementAlgorithm fifo = new FIFO(new Integer(args[1]).intValue());
                for (int i = 0; i < referenceString.length; i++) {
                        fifo.insert(referenceString[i]); }
                // report the total number of page faults
                System.out.println("FIFO faults = " + fifo.getPageFaultCount() + "\n");
                // LRU Algorithm
                System.out.println("LRU Algorithm\n");
                ReplacementAlgorithm lru = new LRU(new Integer(args[1]).intValue(), referenceString);
                for (int i = 0; i < referenceString.length; i++) {
                        lru.insert(referenceString[i]);  }
                // report the total number of page faults (LRU)
                System.out.println("LRU faults = " + lru.getPageFaultCount() + "\n"); }}
```

Output:

```
Output - JavaApplication10 (run)
run:
Reference string = 7 7 0 4

FIFO Algorithm

1 - 7 (pf)
2 - (empty)
3 - (empty)
4 - (empty)
5 - (empty)

1 - 7
2 - (empty)
3 - (empty)
4 - (empty)
5 - (empty)

1 - 7
2 - 0 (pf)
3 - (empty)
4 - (empty)
5 - (empty)

1 - 7
2 - 0
3 - 4 (pf)
4 - (empty)
5 - (empty)

FIFO faults = 3
```

```
Output - JavaApplication10 (run)
LRU Algorithm

1 - 7 (pf)
2 - (empty)
3 - (empty)
4 - (empty)
5 - (empty)

1 - 7
2 - (empty)
3 - (empty)
4 - (empty)
5 - (empty)

1 - 7
2 - 0 (pf)
3 - (empty)
4 - (empty)
5 - (empty)

1 - 7
2 - 0
3 - 4 (pf)
4 - (empty)
5 - (empty)

LRU faults = 3

BUILD SUCCESSFUL (total time: 0 seconds)
```

Practical 8

Using Worker thread write Android code for a click listener that downloads an image from a separate thread and displays it in an ImageView.

Code:

MainActivity.java

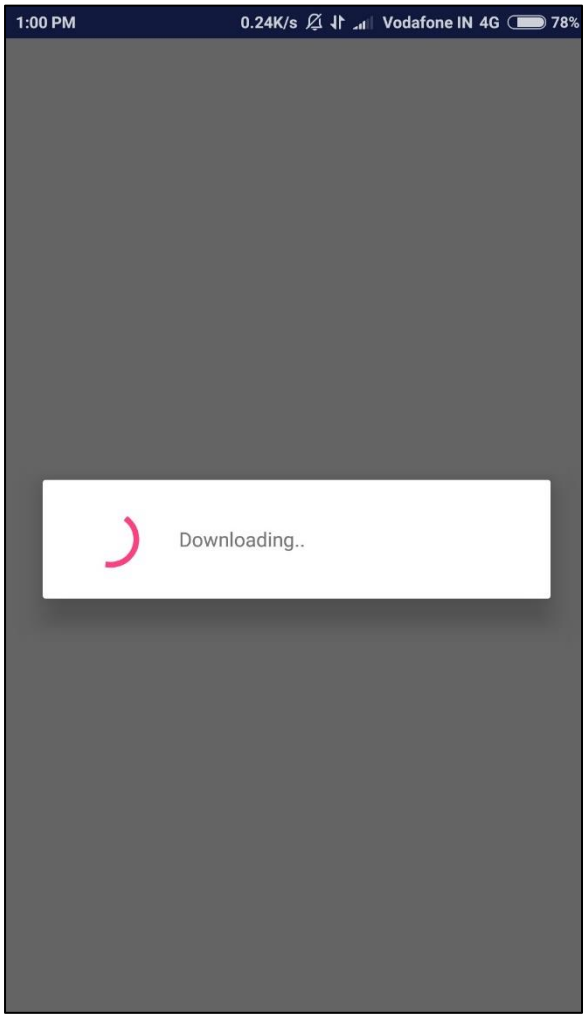
```
package com.example.student.practical8;

import android.app.Activity;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
public class MainActivity extends Activity {
   ImageView image_area;
   ProgressDialog pDialog;
   Bitmap bitmap;
   String imageURL = "http://www.ruiacollege.edu/images/banner/welcome.jpg";
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      image_area = (ImageView) findViewById(R.id.image_xml);
      new download_image().execute();
   }
   public class download_image extends AsyncTask<Void, Void, Void> {
      @Override
      protected void onPreExecute() {
         super.onPreExecute();
         pDialog = new ProgressDialog(MainActivity.this);
         pDialog.setMessage("Downloading..");
         pDialog.setCancelable(false);
         pDialog.show();
      }
      @Override
      protected Void doInBackground(Void... params) {
         try {
            InputStream input = new java.net.URL(imageURL).openStream();
            bitmap = BitmapFactory.decodeStream(input);
         } catch (IOException e) {
            Log.e("Error: ", "" + e);
         }         return null;      }
```

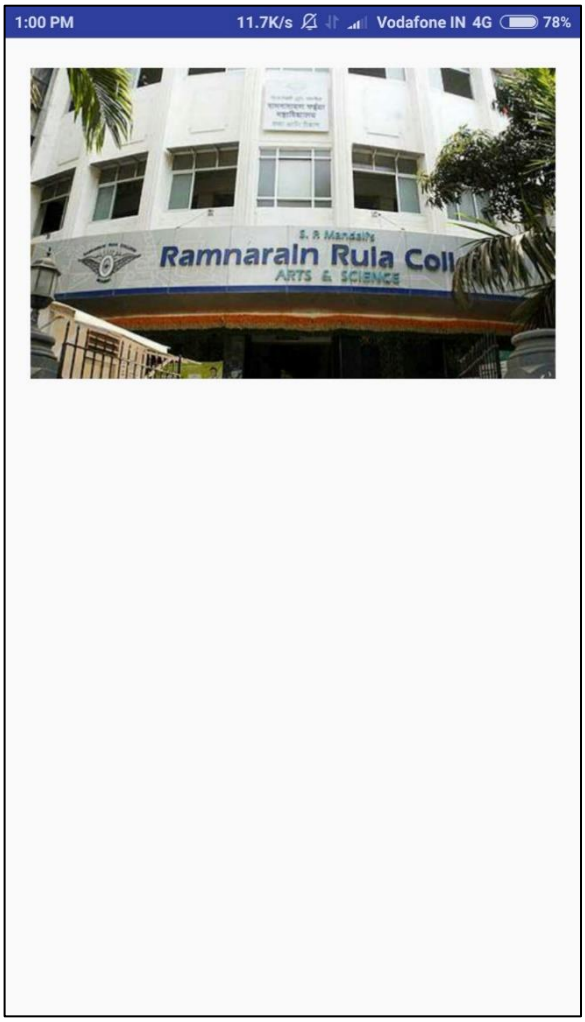
```
    protected void onPostExecute(Void aVoid) {
        super.onPostExecute(aVoid);
        image_area.setImageBitmap(bitmap); pDialog.dismiss();}}}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.student.practical8">
    <uses-permission android:name="android.permission.INTERNET"/>  <!-- //Permission to access the internet
-->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Output:


1:00 PM
0.24K/s
Vodafone IN 4G
78%
Downloading..


1:00 PM
11.7K/s
Vodafone IN 4G
78%
Ramnarain Rula Coll
ARTS & SCIENCE

Practical 9

Write Android activity that includes each of the fundamental lifecycle methods

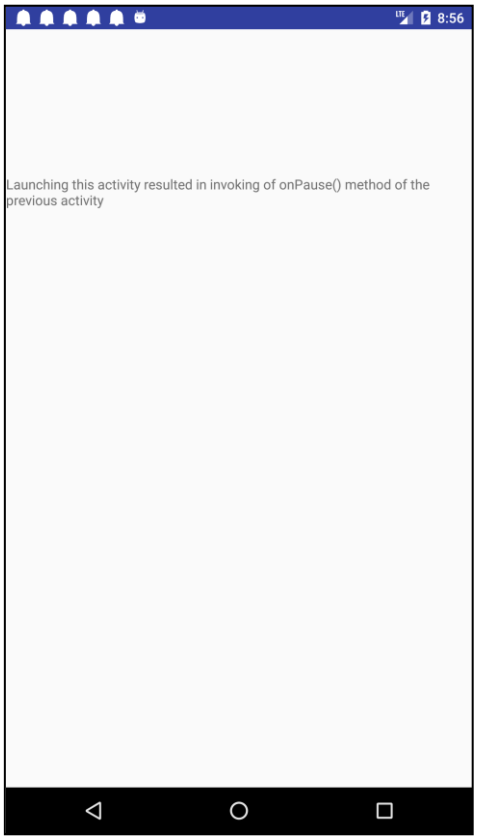
Code:

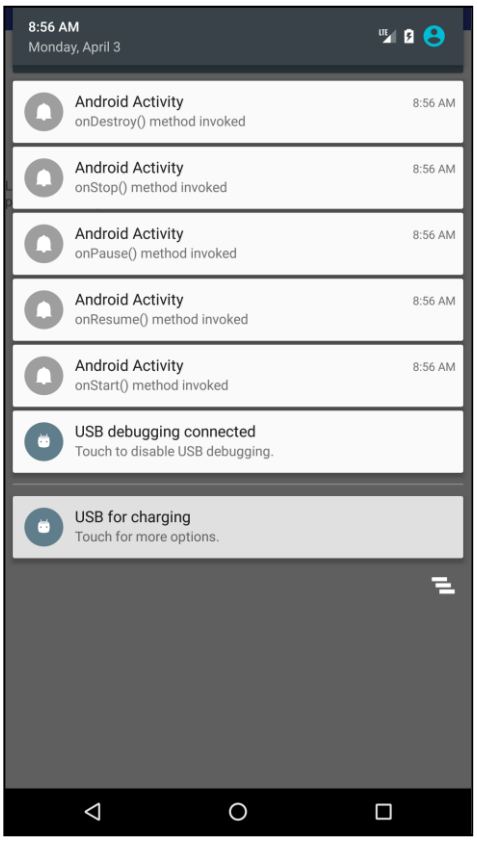
MainActivity.java

```
package practical10.msc.practical9;

import android.app.Activity;
import android.app.NotificationManager;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
   Button pause_btn,stop_btn;
   int mNotificationId = 1;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      pause_btn=(Button) findViewById(R.id.pause);
      stop_btn=(Button) findViewById(R.id.stop);
      pause_btn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            Intent intent = new Intent(MainActivity.this, Empty.class);
            startActivity(intent);}});
      stop_btn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            finish();
         }
      });}
  @Override
   public void onStart() {
      super.onStart();
      Display_Notification("onStart() method invoked");
   }
   @Override
   public void onPause() {
      super.onPause();
      Display_Notification("onPause() method invoked");
   }
   @Override
   public void onResume() {
      super.onResume();
      Display_Notification("onResume() method invoked");
   }
   @Override
```

```
public void onStop() {
   super.onStop();
   Display_Notification("onStop() method invoked");
}
@Override
public void onDestroy() {
   super.onDestroy();
   Display_Notification("onDestroy() method invoked");
}
void Display_Notification(String msg) {
   NotificationCompat.Builder mBuilder =
         new NotificationCompat.Builder(this)
               .setSmallIcon(R.drawable.notification_icon)
               .setContentTitle("Android Activity")
               .setContentText(msg);
   NotificationManager mNotifyMgr =(NotificationManager) getSystemService(NOTIFICATION_SERVICE);
   mNotifyMgr.notify(mNotificationId++, mBuilder.build());}}
```

Output:





Practical 10

Write Android application to demonstrate data storage with following options (any one can be asked in Practical examination):
Shared Preferences (Store private primitive data in key-value pairs)
Internal Storage (Store private data on the device memory)
External Storage (Store public data on the shared external storage)
SQLite Databases (Store structured data in a private database)
Network Connection (Store data on the web with your own network server).
Code:
MainActivity.java

```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import java.util.ArrayList;
import java.util.List;

public class MainActivity extends Activity {
   public EditText name, email;
   Spinner spinner;
   Button save_btn;
   String save_type;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      name = (EditText) findViewById(R.id.f_name);
      email = (EditText) findViewById(R.id.email_xml);
      save_btn = (Button) findViewById(R.id.save_btn);
      spinner = (Spinner) findViewById(R.id.spinner_xml);
      fill_spinner();
      save_btn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            save_type = spinner.getSelectedItem().toString();
            new
Task_Handler(MainActivity.this,name.getText().toString(),email.getText().toString(),save_type).execute();
         }});}
   public void fill_spinner() {
      List<String> list = new ArrayList<String>();
      list.add("SQLite Storage");
      list.add("Internal Storage");
      list.add("External Storage");
      list.add("Shared Preferences");
      list.add("Network Connection");
      ArrayAdapter<String> dataAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, list);
```

```
    dataAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner.setAdapter(dataAdapter);
  }
```

SQLite_DB_Management.java

```
import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class SQLite_DB_Management extends SQLiteOpenHelper{

  public static final String DATABASE_NAME = "student.db";                    //Database name
  public static final String TABLE = "student_data";                          //Table name
  public static final String FNAME_COL = "FNAME";
  public static final String EMAIL_COL = "EMAIL";

  public SQLite_DB_Management(Context context){
    super(context, DATABASE_NAME,null,1);
  }
  @Override
  public void onCreate(SQLiteDatabase db) {
    db.execSQL("CREATE TABLE " + TABLE + "(FNAME TEXT,EMAIL TEXT)");
  }
  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
  }
  public boolean insertData(String name, String email) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(FNAME_COL, name);
    contentValues.put(EMAIL_COL,email);
    long result = db.insert(TABLE, null, contentValues);
    if (result == -1) {
      return false;
    } else {
      return true;}}}
```

Task_Handler.java

```
import android.Manifest;
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.os.*;
import android.support.v4.app.ActivityCompat;
import android.util.Log;
import android.widget.Toast;
import java.io.*;
import java.net.*;
```

```
public class Task_Handler extends AsyncTask<Void, Void, Void> {
   private static final int REQUEST_EXTERNAL_STORAGE = 1;
   private static String[] PERMISSIONS_STORAGE = {Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.WRITE_EXTERNAL_STORAGE};
   ProgressDialog pDialog;
   int flag;
   String name, email, save_type;
   private Context context;
   public Task_Handler(Context cxt, String name, String email, String save_type) {
      context = cxt;
      pDialog = new ProgressDialog(context);
      this.name = name;
      this.email = email;
      this.save_type = save_type;
   }
   @Override
   protected void onPreExecute() {
      super.onPreExecute();
      pDialog = new ProgressDialog(context);
      pDialog.setMessage("Saving..");
      pDialog.setCancelable(false);
      pDialog.show();
   }
   @Override
   protected Void doInBackground(Void... params) {
      switch (save_type) {
         case "SQLite Storage":
            SQL_storage();
            break;
         case "Internal Storage":
            Int_Storage();
            break;
         case "External Storage":
            Ext_Storage();
            break;
         case "Shared Preferences":
            Shared_Preference();
           break;
         case "Network Connection":
            Network_Storage();
            break;
      }
      return null;
   }
   @Override
   protected void onPostExecute(Void aVoid) {
      super.onPostExecute(aVoid);
      pDialog.dismiss();
      if (flag == 1)
         Toast.makeText(context, "Error", Toast.LENGTH_LONG).show();
```

```
        else {
            Toast.makeText(context, "Saved", Toast.LENGTH_LONG).show();
        }
    }
    void SQL_storage() {
        SQLite_DB_Management sql = new SQLite_DB_Management(context);
        sql.insertData(name, email);
    }
    void Int_Storage() {
        String fileName = "Internal_Storage";
        String content = name + "\t" + email;
        FileOutputStream outputStream;
        try {
            outputStream = context.openFileOutput(fileName, Context.MODE_PRIVATE);
            outputStream.write(content.getBytes());
            outputStream.close();
        } catch (Exception e) {
            flag = 1;
            e.printStackTrace();}}
    void Ext_Storage() {
        String content = name + "\t" + email, file_name = "ext_file";
        File file;
        try {
            int permission = ActivityCompat.checkSelfPermission(context,
Manifest.permission.WRITE_EXTERNAL_STORAGE);
            if (permission != PackageManager.PERMISSION_GRANTED) {
                // We don't have permission so prompt the user
                ActivityCompat.requestPermissions((Activity) context, PERMISSIONS_STORAGE,
REQUEST_EXTERNAL_STORAGE);
            }
            file = new File(Environment.getExternalStorageDirectory().getAbsoluteFile(), file_name);
            FileOutputStream fos = new FileOutputStream(file);
            fos.write(content.getBytes());
            fos.close();
        } catch (IOException e) {
            flag = 1;
            e.printStackTrace();}}
    void Shared_Preference(){
        SharedPreferences sharedPreferences =
context.getSharedPreferences("shared_preference_file",Context.MODE_PRIVATE);
        SharedPreferences.Editor editor=sharedPreferences.edit();
        editor.putString("name",name);
        editor.putString("email", email);
        editor.apply();
        if(sharedPreferences.getString("name","").isEmpty() ){
            flag=1;
        }else{
            flag=0;
        }
    }
    void Network_Storage(){
```

```
    String network_storage="http://192.168.0.104/save.php";
    try {
      URL url=new URL(network_storage);
      HttpURLConnection httpURLConnection=(HttpURLConnection)url.openConnection();
      httpURLConnection.setRequestMethod("POST");
      httpURLConnection.setDoOutput(true);
      httpURLConnection.setDoInput(true);
      OutputStream outputStream=httpURLConnection.getOutputStream();
      BufferedWriter bufferedWriter=new BufferedWriter(new OutputStreamWriter(outputStream,"UTF-8"));
      String post_data= URLEncoder.encode("name", "UTF-8")+"="+URLEncoder.encode(name,"UTF-8")+"&"
          +URLEncoder.encode("email", "UTF-8")+"="+URLEncoder.encode(email,"UTF-8");
      bufferedWriter.write(post_data);
      bufferedWriter.flush();
      bufferedWriter.close();
      outputStream.close();
      InputStream inputStream=httpURLConnection.getInputStream();
      BufferedReader bufferedReader=new BufferedReader(new InputStreamReader(inputStream,"iso-8859-
1"));

      String result="",line="";
      while ((line=bufferedReader.readLine())!=null){
        result+=line;
      }
      bufferedReader.close();
      inputStream.close();
      httpURLConnection.disconnect();
      Log.e("RESULT:","+result);
    }catch (Exception e){
      Log.e("ERROR:","+e);}}}
```

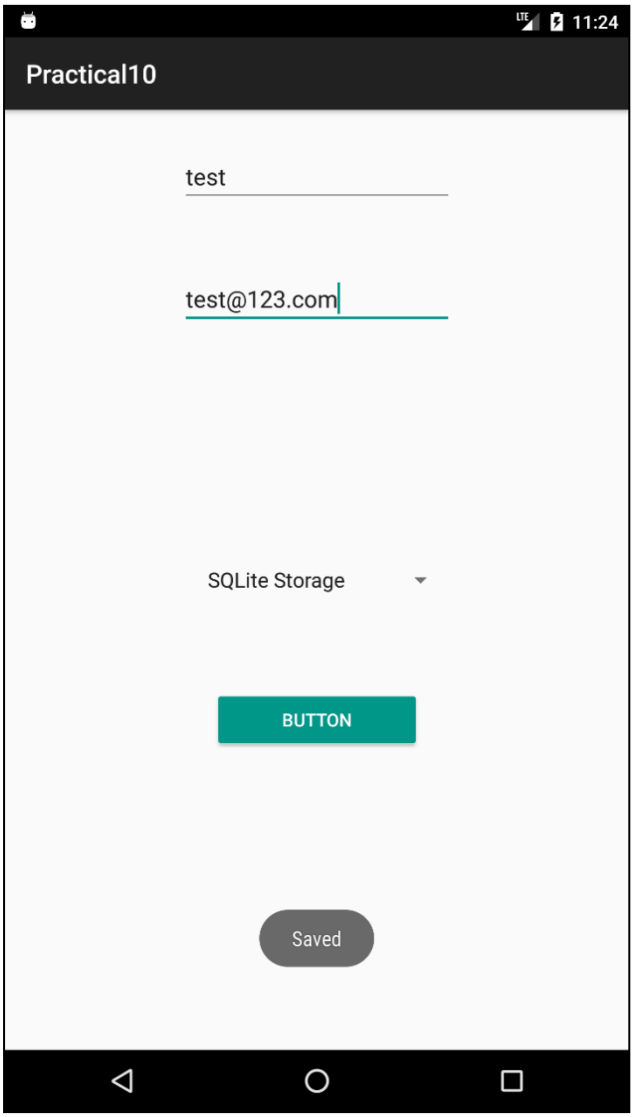
AndroidManifest.xml

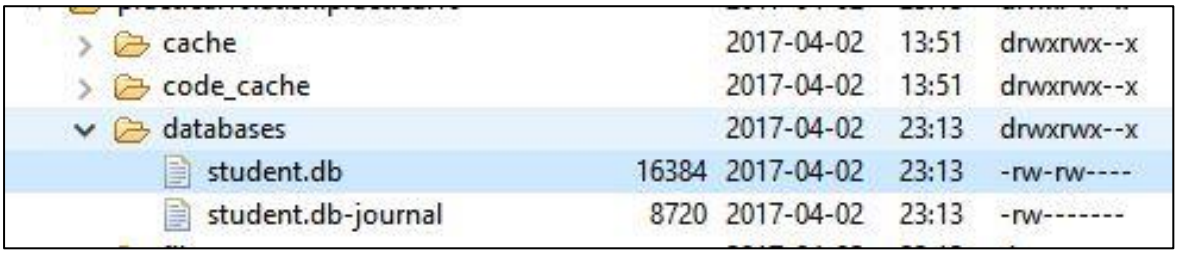
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="practical10.atish.practical10">
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.INTERNET" />
```

```
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
</manifest>
```

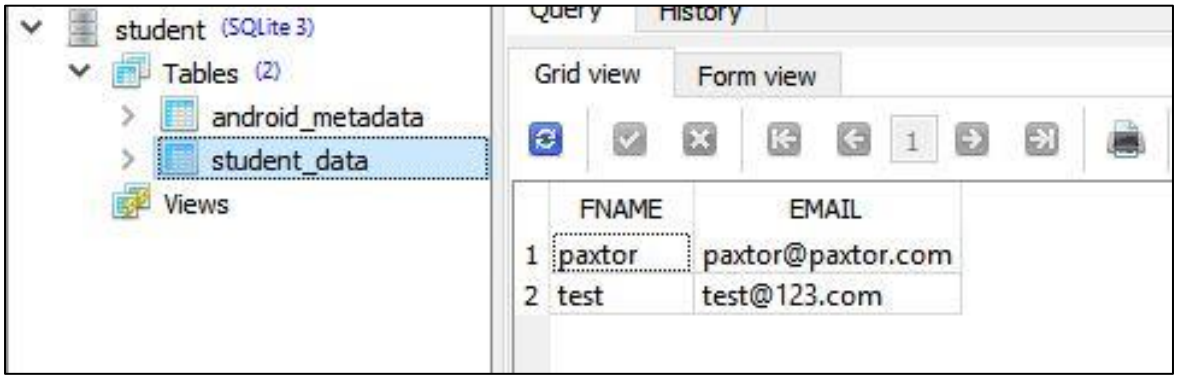
Output:

SQLite Storage

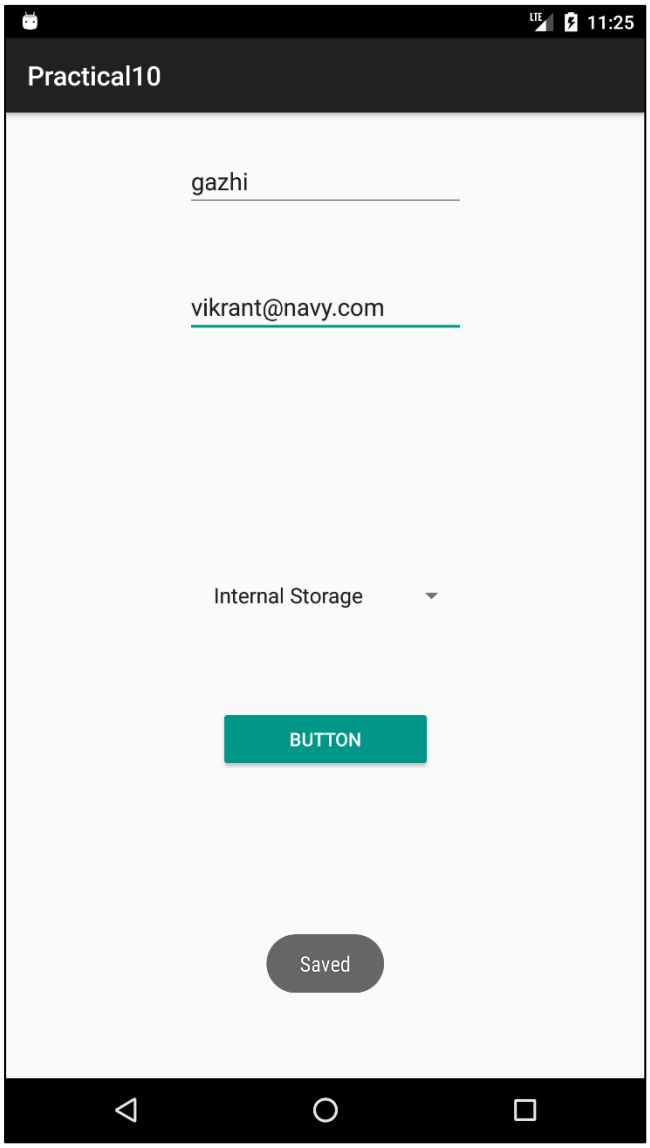


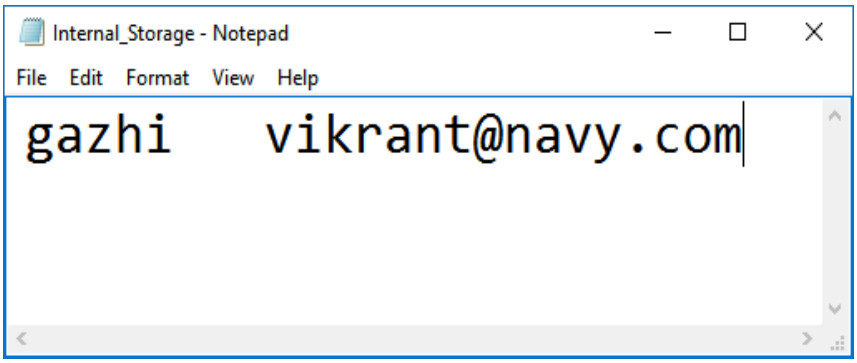


cache		2017-04-02	13:51	drwxrwx--x
code_cache		2017-04-02	13:51	drwxrwx--x
databases		2017-04-02	23:13	drwxrwx--x
student.db	16384	2017-04-02	23:13	-rw-rw----
student.db-journal	8720	2017-04-02	23:13	-rw-------

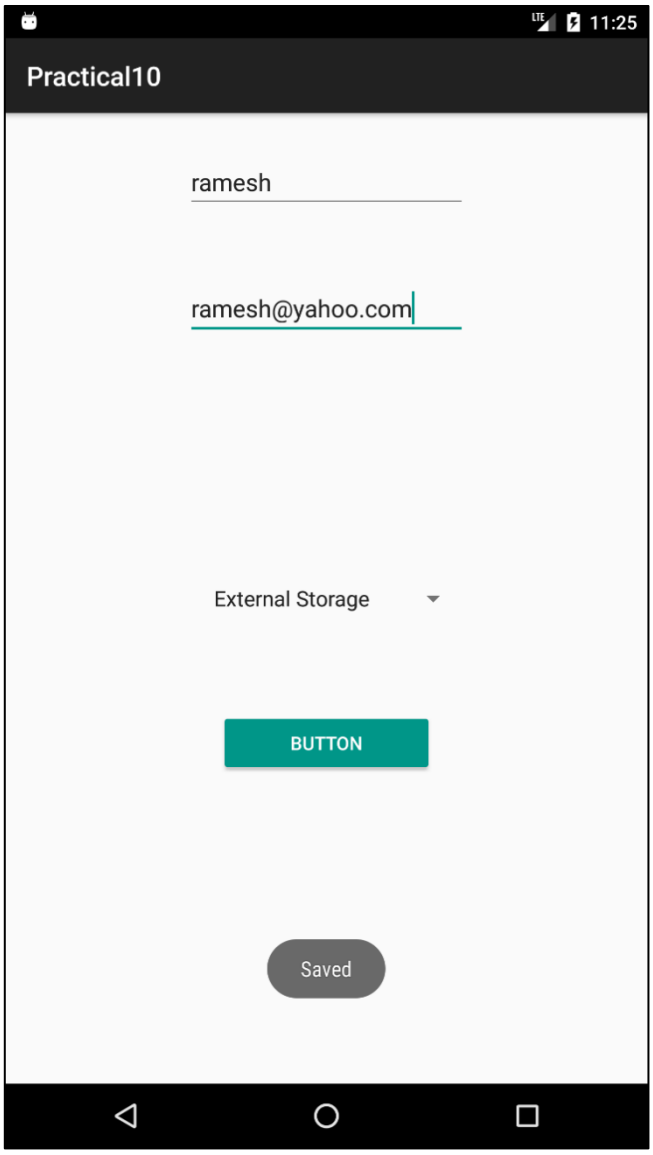


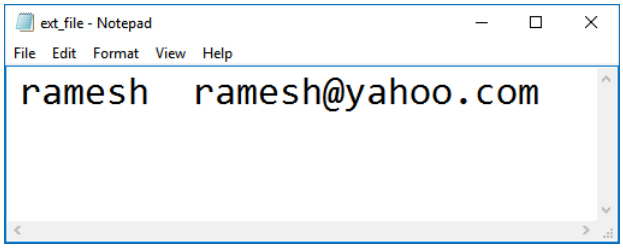
Internal Storage

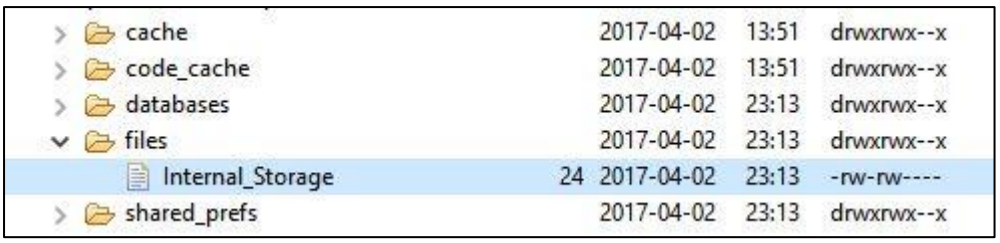




External Storage



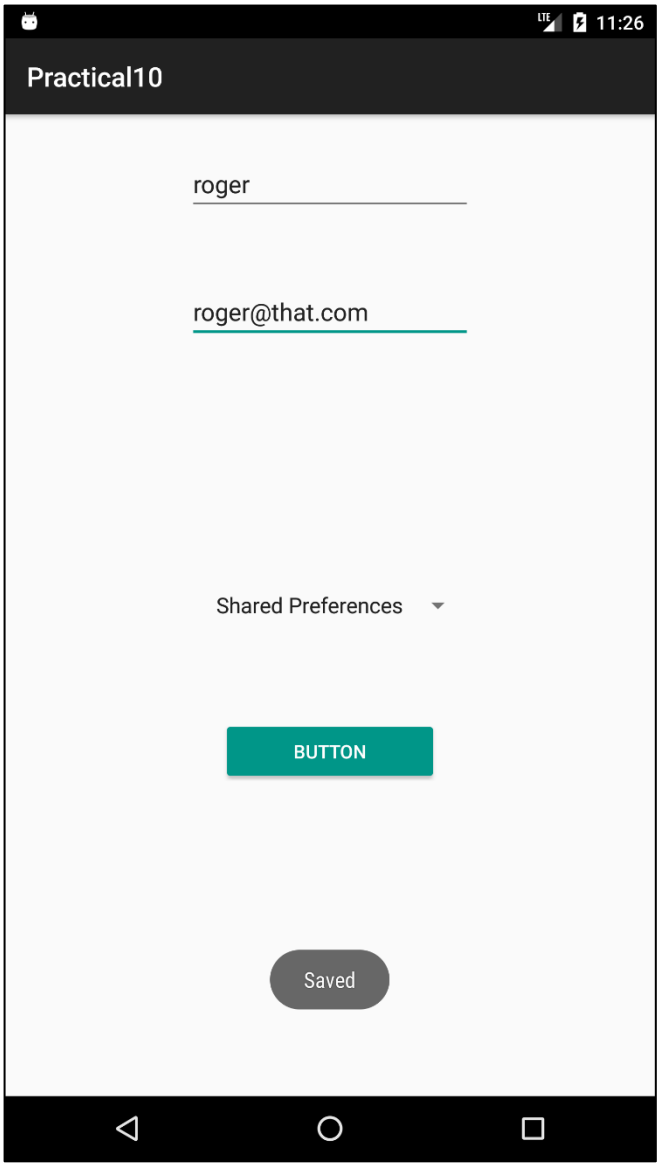




cache		2017-04-02	13:51	drwxrwx--x
code_cache		2017-04-02	13:51	drwxrwx--x
databases		2017-04-02	23:13	drwxrwx--x
files		2017-04-02	23:13	drwxrwx--x
Internal_Storage	24	2017-04-02	23:13	-rw-rw----
shared_prefs		2017-04-02	23:13	drwxrwx--x

storage		2017-04-02	23:04	drwxr-xr-x
0F03-2D02		1970-01-01	05:30	drwxrwx--x
emulated		2017-04-02	13:05	drwx--x--x
0		2017-04-02	15:19	drwxrwx--x
Alarms		2017-04-02	13:05	drwxrwx--x
Android		2017-04-02	13:05	drwxrwx--x
DCIM		2017-04-02	15:01	drwxrwx--x
Download		2017-04-02	13:05	drwxrwx--x
Movies		2017-04-02	13:05	drwxrwx--x
Music		2017-04-02	13:05	drwxrwx--x
Notifications		2017-04-02	13:05	drwxrwx--x
Pictures		2017-04-02	13:05	drwxrwx--x
Podcasts		2017-04-02	13:05	drwxrwx--x
Ringtones		2017-04-02	13:05	drwxrwx--x
ext_file	23	2017-04-02	23:25	-rw-rw----

Shared Preferences



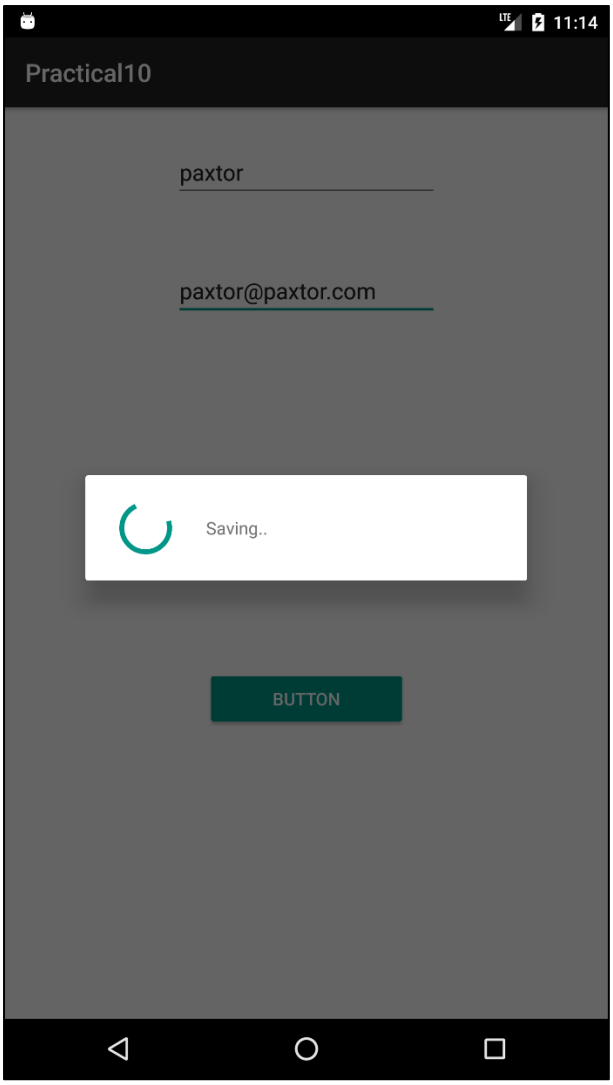
cache		2017-04-02	13:51	drwxrwx--x
code_cache		2017-04-02	13:51	drwxrwx--x
databases		2017-04-02	23:13	drwxrwx--x
files		2017-04-02	23:13	drwxrwx--x
shared_prefs		2017-04-02	23:13	drwxrwx--x
shared_preference_file.xml	162	2017-04-02	23:13	-rw-rw----

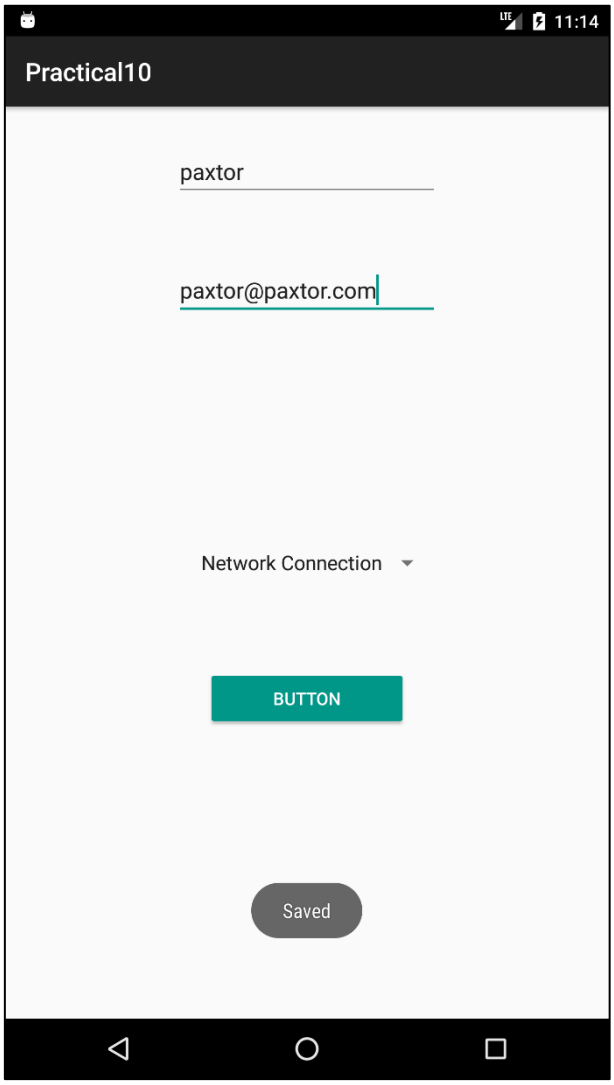
shared_preference_file.xml - Notepad

File Edit Format View Help

```
<?xml version='1.0' encoding='utf-8'
standalone='yes' ?>
<map>
<string name="email">roger@that.com</string>
<string name="name">roger</string>
</map>
```

Network Storage





```
root@Davy: ~
File  Edit  View  Search  Terminal  Help
Server version: 5.6.30-1 (Debian)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use AOS;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from app_data;
+--------+-------------------+
| name   | email             |
+--------+-------------------+
| paxtor | paxtor@paxtor.com |
+--------+-------------------+
1 row in set (0.00 sec)

mysql>
```