public class PlayList {

Q.1(a)**new ArrayList<Song>()**

private ArrayList<Song> playList = ; **//1 Mark**

public void addSong(String strSong){

Song song = null;

Q.1(b) **indexOf(",")**

int index = strSong. ; **//2 Marks**

Q.1 (c) **substring**

String type = strSong. (0,index); **//2 Marks**

Q.1 (d) **equalsIgnoreCase**

if("Mp3". (type)){ **//2 Marks**

Q.1(e) **song = new Mp3(strSong);**

**//2 Marks**

Q.1(f) **equalsIgnoreCase**

}else if("Vid". (type)){ **//2 Marks**

Q.1(g) **song = new VidSong(strSong);**

**//2 Marks**

}else{ System.out.println("Illegal Song Type"); }

/\* write code to add a song to the playlist. Duplicate entry

of the same song is not allowed use checkIfSongExisting()

method to disallow duplicate \*/

Q.1(h) **!checkIfSongExisting(song)**

if( ){

Q.1 (i) **playList.add(song);**

} **//2 Marks**

} //End of addSong() method

private boolean checkIfSongExisting(Song newSong){

Q.1 (j) **Iterator itr = playList.iterator();**

**while(itr.hasNext()){**

**if(newSong.equals(itr.next()) return true;**

**}**

**return false; //2 Marks**

} //End of checkIfSongExisting() method

Q.1 (k) **ArrayList<Song>**

public getSortedList(){ **//1 Mark**

Q.1 (l) **Collections.sort(playList, new Comparator<Song>() {**

**public int compare(Song s1, Song s2) {**

**if(s1.yrOfRel < s2.yrOfRel) return -1;**

**if(s1.yrOfRel > s2.yrOfRel) return 1;**

**return 0;**

**}}); //9 Marks**

return playList; } // End of getSortedList() method

public String toString(){

//code to print the elements of the list in this method

Q.1 (m) **StringBuffer sb = new StringBuffer();** **//1 Mark**

for (Song sd : playList) {

Q.1(o) **toString()**

Q.1 (n) **append**

sb. (sd. + "\n"); //**2 Marks**

} //End of for loop

Q.1 (p) **toString**

return sb. ; /**/1 Mark**

} //End of toString() method

/\*\*\*\*\*\*\*\*\* Abstract Inner class Song \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Q.2(a) **abstract class Song implements Comparable<Song>** **//1 Mark**

private {

private String sName;

protected String type;

private int yrOfRel;

protected int audioLen;

public Song(String sDes){

Q.2(b) **StringTokenizer st = new StringTokenizer(sDes, ",");**

**this.type = st.nextToken();**

**this.sName = st.nextToken();**

**this.yrOfRel = Integer.parseInt(st.nextToken());**

**this.audioLen = Integer.parseInt(st.nextToken());**

**//5 Marks**

}

Q.2(c) **public int compareTo(Song song) { //3 Marks**

**if(this.audioLen < song.audioLen)**

**return -1;**

**if(this.audioLen > song.audioLen)**

**return 1;**

**return 0;**

**}**

Q.2(d) **public boolean equals(Song song){ //3 Marks**

**if(this.type.equalsIgnoreCase(song.type)**

**&& this.sName.equalsIgnoreCase(song.sName))**

**return true;**

**return false;**

**}**

public String toString() {

return "Song [sName=" + sName + ", type=" + type + ", yrOfRel=" + yrOfRel + "]";

}

}//End of abstract class Song

/\*\*\*\*\*\*\*\*\* Concrete Inner class Mp3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private class Mp3 extends Song {

Q.3 (a) **super(sDes);**

public Mp3(String sDes) { } **//1 Mark**

}//End of inner class Mp3

/\*\*\*\*\*\*\*\*\*\* Concrete Inner class VidSong \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private class VidSong extends Song {

private int videoLen;

public VidSong(String sDes) {

super(sDes);

Q.3 (b) **super(sDes); // 3 Marks**

**int index = sDes.lastIndexOf(",");**

**String vidLen = sDes.substring(index+1);**

**this.videoLen = Integer.parseInt(vidLen);**

}

Q.3 (c) **public int compareTo(Song song) {**

**if(song instanceof VidSong){**

**VidSong vidS = ((VidSong)song);**

**if(this.audioLen < song.audioLen**

**&& this.videoLen < vidS.videoLen)**

**return -1;**

**if(this.audioLen > song.audioLen**

**&& this.videoLen > vidS.videoLen) return 1;**

**}**

**return 0;**

**}**

**}**

}//End of inner class VidSong

}//End of class PlayList

\*\*\*\*\*\*\*\*\*\*\*\*\* BEST OF LUCK \*\*\*\*\*\*\*\*\*\*\*\*\*\*