Exercise - 1

①
```
Public class DooBee {
  Public class static void main (String [] args)
  {
     int x = 1 ;
     while ( x < 3)
     {
        System.out.Print(" Doo ");
        System.out.Print (" Bee ");
        x++;
     }
     if (x == 3)
     {
        System.out.Print ("Do");
     }
  }
}
```

② Code Magnets

```
Class Shuffle1 {
  Public static void main (String args[])
  {
     int x = 3 ;
     while ( x > 0)
     {
        if (x > 2)
        {
           System.out.Print ("a");
        }
```

∴ System.out.Print(" "); = S.O.U.t

x--;
System.out.Print("_ ");

if(x==2)
{
S.O.U.t(" b c");
}

if(x==1)
{
S.O.U.t("d");
x--;
}
}
}
}

O/P :- a-b c-d

(3) Be the Compiler.

(A)
class Exercise1a
{
Public static void main(String[]args)
{
int x=1;
while(x<10)
{
x++;
if(x>3)
System.out.println(" big x");
}
}
}

(b)

```
class Exercise1b
{
    Public static void main (string [] args)
    {
        int x=5;
        while (x>1)
        {
            x--;
            if (x<3)
            System.out.println(" small x ");
        }
    }
}
```

(a)

```
class Exercise1c
{
    Public static void main (string [] args)
    {
        int x=5; while (x>1){
        x--;
        if (x<3)
        {
            System.out.println(" small x ");
        }
    }
}
```

(4) Pool Puzzle

```
class PoolPuzzleOne
{
    public static void main(String [] args)
    {
        int x = 0;

        while (x < 4)
        {
            System.out.print("a");

            if (x < 1)
            {
                System.out.print(" ");
            }
            System.out.print("n");

            if (x > 1)
            {
                System.out.print("ystem");
                x = x + 2;
            }
            if (x == 1)
            {
                System.out.print("nows");
            }
            if (x < 1)
            {
                System.out.print("oise");
            }
            System.out.println();
            x++;
        }
    }
}
```

(5) Mixed Messages

```
class Test
{
    public static void main(String args[])
    {
        int x=0;
        int y=0;
        while(x<5)
        {

            System.out.Print(x+" "+y+" ");
            x++;
        }
    }
}
```

Candidates

(a) `y=x-1;`
(b) `y=y+x;`
(c) `y=y+2;`
`if(y>4)`
`{`
`y=y-1;`
`}`
(d) `x=x+1;`
`y=y+x;`
(e) `if(y<5){`
`x=x+1;`
`if(y<3){`
`x=x-1;`
`}`
`y=y+2;`

Possible outputs

00 11 21 32 42
00 11 23 36 410
02 14 25 36 47
11 13 34 59
02 14 36 48

Exercise - 2

(1) Be the Compiler

(A)

```
class StreamingSong
{
    String title;
    String artist;
    int duration;

    void play()
    {
        System.out.println("Playing song");
    }

    void printDetails()
    {
        System.out.println("This is " + title +
            " by " + artist);
    }
}

class StreamingSongTestDrive
{
    public static void main(String[] args)
    {
        StreamingSong song = new StreamingSong();
        song.artist = "The Beatles";
        song.title = "Delivery";
        song.play();
        song.printDetails();
    }
}
```

O/P : This is Bdrawn by The Beatler

(15)
```
class Episode
{
    int seriesNumber;
    int episodeNumber;

    void play()
    {
      System.out.Println("Playing episode" + episodeNumber);
    }

    void skipIntro()
    {
      System.out.Println("skipping intro....");
    }

    Void skipToNext()
    {
      System.out.Println("Loading next episode...");
    }
}

class EpisodeTestDrive
{
    Public static void main(String[] args)
    {
      Episode episode = new Episode(); episode.seri-
esNumber=4; episode.play();
      episode.skipIntro();

} }
```

(2) Code Magnet:

```
class Drumkit
{
    boolean topHat = true;
    boolean snare = true;

    void playTopHat()
    {
        System.out.println("ding ding da-ding");
    }

    void playSnare()
    {
        System.out.println("bang bang ba-bang");
    }
}

class DrumkitTestDrive
{
    public static void main(String[] args)
    {
        Drumkit d = new Drumkit();
        d.playSnare();
        d.snare();
        d.playTopHat();

        if(d.snare == true)
        {
            d.playSnare();
        }
    }
}
```

(3) Pool Puzzle

(a)
```
Public class EchoTestDrive
{
    Public static void main (String [] args)
    {
        Echo e1 = new Echo ();
        Echo e2 = new Echo()
                or
        Echo e2 = e1

        int x = 0;
        while (x < 4)
        {
            e1.hello();
            e1.count = e1.count + 1;
            if (x == 3)
            {
                e2.count = e2.count + 1;
            }
            if (x > 0)
            {
                e2.count = e2.count + e1.count;
            }
            x++;
        }
        System.out.println (e2.count);
    }
}
```

(1)

```
class Echo
{
    int count = 0;

    void hello()
    {
        System.out.println("hello000...");
    }
}
```

Exercise – 3

(1) Be the Compiler

(A)

```
class Books
{
    String title;
    String author;
}

class BookTestDrive
{
    public static void main (String args[])
    {
        Books [] myBooks = new Books[3];
        int x=0;
        myBooks[0] = new Books();
        myBooks[1] = new Books();
        myBooks[2] = new Books();

        myBooks[0].title = "The Grapes of Java";
        myBooks[1].title = "The Java Gatsby";
        myBooks[2].title = "The Java Cookbook";
        myBooks[0].author = "bob";
        myBooks[1].author = "sue";
        myBooks[2].author = "ian";
        while (x < 3)
        {
            System.out.Print(myBooks[x].title);
            System.out.Print("by");
            System.out.println(myBooks[x].author);
        } x++;
    }
}
```

(13)

```
class Hobbit
{
    String name;

    public static void main(String[] args)
    {
        Hobbit[] h = new Hobbit[3];

        int z = -1;
        while (z<2)
        {
            z = z+1;
            h[z] = new Hobbit();
            h[z].name = "bilbo";
            if (z == 1)
            {
                h[z].name = "frodo";
            }
            if (z == 2)
            {
                h[z].name = "sam";
            }

            System.out.print(h[z].name + "
                is a");
            System.out.println("good Hobbit name");
        }
    }
}
```

(2) Code Magnets

```
class TestArrays
{
    Public static void main(String [] args)
    {
        int [] index = new int [4];
        index[0] = 1 ;
        index[1] = 3 ;
        index[2] = 0 ;
        index[3] = 2 ;
        String[] islands = new String [4];

        Islands [0] = "Bermuda" ;
        Islands [1] = "Fiji" ;
        Islands [2] = "Azores" ;
        Islands [3] = "Cozumel" ;
        int y = 0 ;

        Int xy ;
        while (y < 4)
        {
            xy = index[y] ;
            System.out.Print(" Island = ");
            System.out.Print(Islands [xy]);
            y ++ ;
        }
    }
}
```

(3) Pool Puzzle

```
class Triangle
{
    double area;
    int height;
    int length;

    public static void main (String [] args)
    {
        int x=0;
        Triangle [] ta = new Triangle [4];
        while (x<4)
        {
            ta[x] = new Triangle ();
            ta[x].height = (x+1)*2;
            ta[x].length = x+4;
            ta[x].setArea();
            System.out.print("triangle " + x +
                ", area");
        System.out.println(" = " + ta[x].area);
        x++;
        }
        int y=x;
        x=27;
        Triangle t5 = ta[2];
        ta[2].area = 343;
        System.out.print("y= " + y);
        System.out.println(", t5 area = "
            + t5.area);
    }
```

```
void setArea()
{
    area = (height * length) / 2;
}
}
```

```
triangle 0, area = 4.0
triangle 1, area = 10.0
triangle 2, area = 18.0
triangle 3, area = 28.0
y = 4, tr area = 343.0
```

Exercise - 4

(1) Be the compiler

(A)

```
class XCopy
{
    public static void main(String[] args)
    {
        int orig = 42;
        XCopy x = new XCopy();
        int y = x.go(orig);
        System.out.println(orig + " " + y);
    }

    int go(int arg)
    {
        arg = arg * 2;
        return arg;
    }
}
```

o/p 42 84

(1)

```
class clock
{
    String time;
    void setTime(String t)
    {
        time = t;
    }

    String getTime()
    {
        return Time;
    }
}

class ClockTestDrive
{
    public static void main(String[] args)
    {
        Clock c = new Clock();
        c.setTime("1245");
        String tod = c.getTime();
        System.out.println("time: " + tod);
    }
}
```

(8) Mixed Messages

```
public class Mix4
{
    int counter = 0;
    public static void main (String[] args)
    {
        int count = 0;
        Mix4[] mixer = new Mix4[20];
        int i = 0;
        while ( [          ] )
        {
            mixer[i] = new Mix4();
            mixer[i].counter = mixer[i].counter + 1;
            count = count + 1;
            count = count + mixer[i].maybeNew(i);
            i++;
        }
        System.out.println(count + " " +
            mixer[1].counter);
    }

    public int maybeNew(int index)
    {
        if ( [          ] )
        {
            Mix4 mix = new Mix4();
            mix.counter = mix.counter + 1;
            return 1;
        }
        return 0;
    }
}
```

Candidates		Possible output
(a)	p < 9 index < 5	14 1
(b)	i < 20 index < 5	25 1
(c)	p < 7 index < 7	14 1
(d)	p < 19 index < 1	20 1

(c) Pool Puzzle

```
Public class puzzle4 {
  public static void main (String args[])
  {
      Value [] values = new Value();
      Int number = 1;
       Int i = 0;
      while (i < 6)
      {
          values [i] = new Value ();
          values [i]. PutValue = number;
          number = number * 10;
          i++;
      }
      Int result = 0;
      i = 6;
      while (i > 0)
      {
        i--;
        result += values [i]. doStuff(i);
      }
```

```
        System.out.println("result" + result);
    }
}

Class Value
{
    int IntValue;

    Public int doStuff (int factor)
    {
        if (IntValue * factor > 100)
            return IntValue * factor;
        else
            return IntValue * (5-factor);
    }
}
```

o/p result 5433.45

Exercise - 5

(A) Be The JVM

```
class OutPut
{
    Public static void main (String [] args)
    {
        Output output = new Output ();
        output.go();
    }

    void go()
    {
        int value = 7;
        for (int i = 1; i < 8; i++)
        {
            value = value + 1;
            if ( i > 4)
            {
                System.out.println (++value + " ");
            }
            if ( value > 14)
            {
                System.out.println (" i = " + i);
                break;
            }
        }
    }
}
```

O/P - 13 15 x = 6

(B) Code Magnet

```
class Multifor
{
    Public static void Main(String[] args)
    {
        for(int i=0; i<4; i++)
        {
            for(int j=4; j>2; j--)
            {
                System.out.println(i+" "+j);
            }

            if(i==1)
            {
                i++;
            }
        }
    }
}
```

O/P - 0 4
0 3
1 4
1 3
3 4
3 3

④ Mixed Messages-

```
Public static void main (String [] args)
{
    int x = 0;
    int y = 30;
    for (int outer = 0; outer < 3; outer++)
    {
        for (int inner = 4; inner > 1; inner--)
        {
            [          ]
            y = y - 2;
            if (x == 6)
                break;
            x = x + 3;
        }
        y = y - 2;
    }
    System.out.println (x + " " + y);
}
```

Candidates	Possible outputs
x = x + 3;	54 6
x = x + 6;	60 10
x = x + 2;	45 6
x++;	36 6
x--;	18 6
x = x * 10;	6 14

Exercise - 6

(A) Code Magnets

```
import java.util.ArrayList;

Public class ArrayListMagnet
{
    public static void main(String[]args)
    {
        ArrayList<String> a = new
            ArrayList<String>();
        a.add(0, "zero");
        a.add(1, "one");
        a.add(2, "two");
        a.add(3, "three");
        printList(a);

        if(a.contains("three"))
        {
            a.add("four");
        }
        a.remove(2);
        printList(a);

        if(a.indexOf("four") != 4)
        {
            a.add(4, "4.2");
        }
        PrintList(a);
```

```
if (a.contains ("two"))
{
    a.add ("2.2");
}
printList (a);
}

public static void printList (ArrayList<String>
                              list)
{
    for (String element : list)
    {
        S.o.u.println (element + " ");
    }
    S.o.u.println ();
}
```

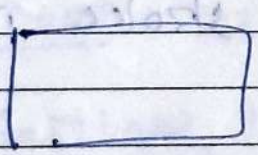
O/P - zero one two three
zero one three four
zero one three four 4.2
zero one three four 4.2

Exercise 7

(A) The program

```
Class A
{
  Int Num= 7;

  void m1()
  {
    System.out("A's m2", 4);
  }
  void m2()
  {
    System.out("A's m2", 4)
  }
    void m3(){
      System.out("A's m3", 4);
    }
}

Class B extends A
{
  void m1()
  {
    System.out("B's m1", 4);
  }
}
```

```
class C extends B
{
    void m3()
    {
        System.out.println("B's C's m3 " + (num+6));
    }
}

Public class Mixed2
{
    Public static void main (String[] args)
    {
        A a = new A();
        B b = new B();
        C c = new C();
        A a2 = new C();
```



```
    }
}
```

Bdc candidates

o/p

(a)
```
b.m1();
c.m2();
a.m3();
```
B's m1, A's m2, A's m3

(b)
```
c.m1();
c.m2();
c.m3();
```
B's m1, A's m2, C's m3

(c) a.m1();
b.m2();
c.m3();
A's m2, A's m2, C's m3 13

(d) a1.M1()
a1.m2();
a1.m3();
B's m2, A's m2, C's m3 13

(13) Pot Pozzle

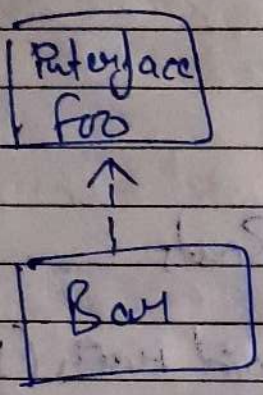
```
Public class RowBoat extends Boat
{
    Public void rowTheBoat()
    {
        S.o.println("stroke natasha");
    }
}

Public class Boat
{
    Private int length;
    Public void setLength(int len)
    {
        length = len;
    }

    Public int getLength()
    {
        return length;
    }
```

```
Public void move()
{
    S.o.u.t("drift");
}
}

Public class TestBoats
{
    Public static void main(String[] args)
    {
        Boat b1 = new Boat();
        Sailboat b2 = new Sailboat();
        Rowboat b3 = new Rowboat();
        b2.setLength(32);
        b1.move();
        b3.move();
        b2.move();
    }
}

Public class Sailboat extends Boat
{
    Public void move()
    {
        S.o.u.t("hoist sail");
    }
}
```

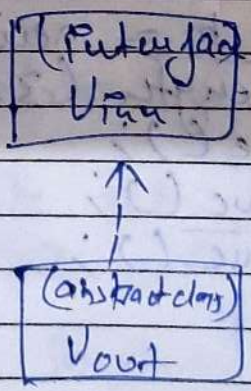
Exercise - 6

(A) what's the picture ?

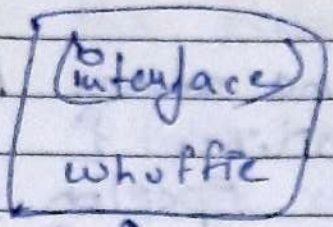
(1) Public Interface Foo { }
Public class Bar implements Foo { }

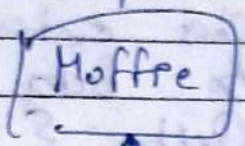


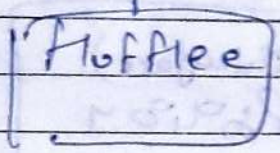
(2) Public Interface Vinn { }
Public abstract class Vout implements Vinn { }



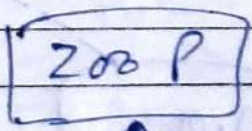
(3) Public abstract class Muffie implements Whuffie { }
Public class fluffie extends Muffie { }
Public interface Whuffie { }

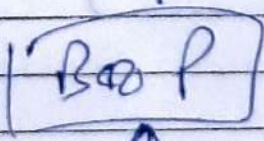


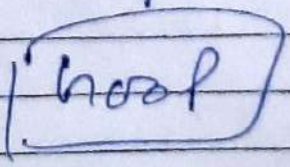




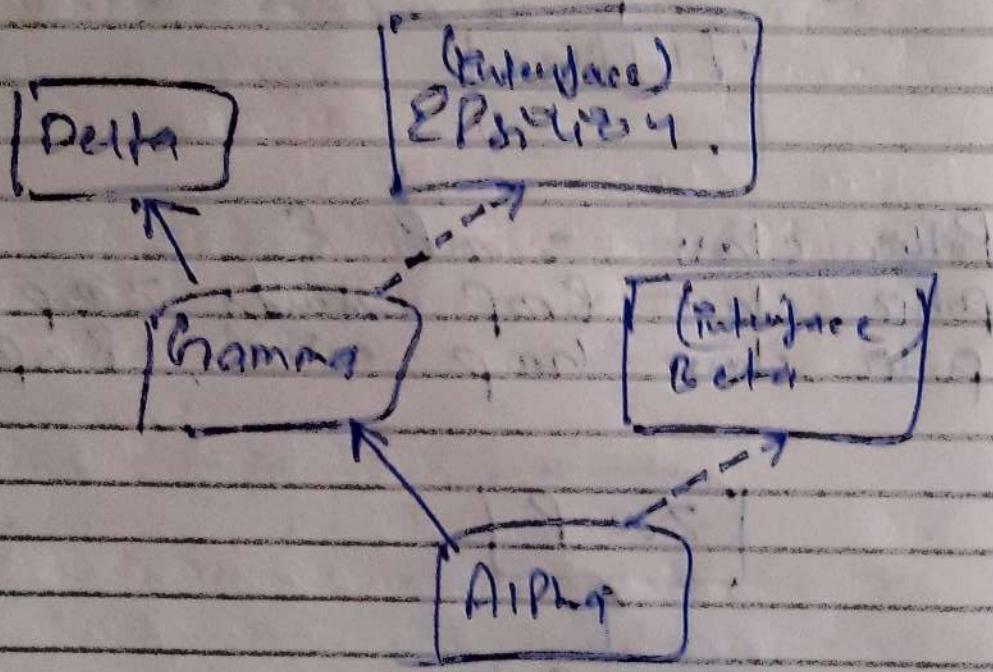
(4) Public class Zoop { }
public class Boop extends Zoop { }
public class Hoop extends Boop { }



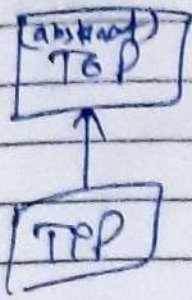




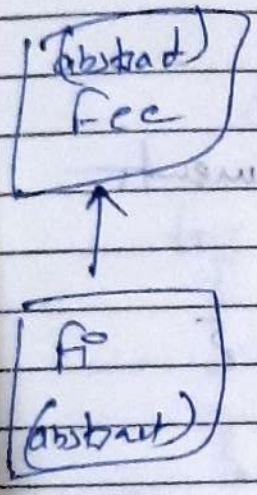
(5) → Public class Gamma extends Delta implements Epsilon {}
→ Public interface Epsilon {}
public interface Beta {}
→ Public class AlPha extends Gamma implements Beta {}
public class Delta {}



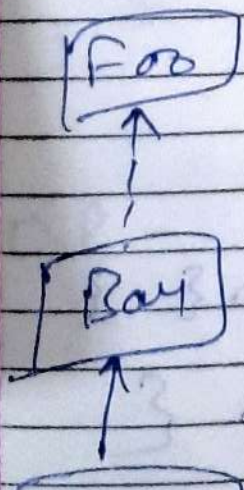
(13) what's the Declaration.

(1)


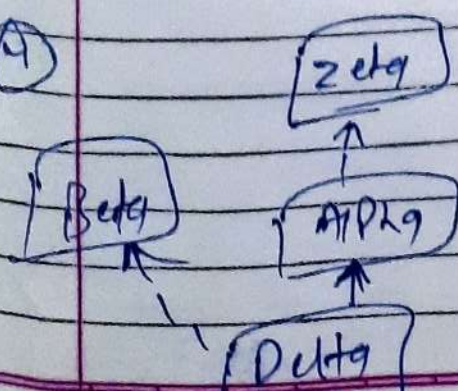
→ Public abstract class TOP { }
→ Public class TIP extends TOP { }

(2)


→ Public abstract class Fee { }
→ public abstract class fi extends Fee { }

(3)


→ Public interface Foo { }
→ Public class Bar implements Foo { }
→ Public class Baz extends Bar { }

(4)


→ Public interface zeta { }
→ Public class Alpha implements zeta { }
→ Public interface Beta { }
→ Public class Delta extends Alpha implements Beta { }

(c) Pol Rozzle

interface Nose
{
Public Int iMethod();
}

abstract class picasso implements Nose {

Public int iMethod() {

return 7

}
}

class Clown extends picasso { }

class Actz extends picasso {
Public int iMethod()
{
return 5;
}
}

```
Public class Of76 extends Clownt
{
    Public static void main (String [] args)
    {
        Norsc [] i = new Norsc [3];
        i[0] = new Acty ();
        i[1] = new Clownt ();
        i[2] = new Of76 ();

        for (int x = 0; x < 3; x++)
        {
            S.o.o.t (i[x].iMethod ()+ " "
            + i[x].getClass ());
        }
    }
}
```

O-P –

```
5 class Acty
7 class Clownt
7 class Of76
```

Exercise - 9

(1) Be the Garbage Collector

1 copyHC = null;

No - this CR attempts to access a variable that is out of scope.

2 - gc2 = null;

Ok - gc2 was the only reference variable referring to that object

3 - newHC = gc3;

No - another out of scope variable.

4 - gc1 = null;

No - newHC is out of scope.

5 - newHC = null;

No - newHC is out of scope;

6 - gc4 = null;

No - gc3 is still referring to that object

7 gc3 = gc2;

No. gc4 still referring to that obj.

8 gc1 = gc4;

Ok - reassigning the only reference to that object

9. gc3 = null

No - gc4 is still referring to that object.

(13) Popular Objects

-> There are total 12 active references to this object right before the main() method completes.
The kit.honeyPot variable is valid for a while, but kit gets nulled at the end since raccoon.rk still refers to the kit object, raccoon.kit.honeyPot (although never explicitly declared)

1- 'honey Pot
2- ha[0]
3- ha[1]
4- ha[2]

5 - ha[3]
6 - bees.beeHoney[0]
7 - bee.beeHoney[1]
8 - bees.beeHoney[2]
9 - bees.beeHoney[3]
10 - bears[0].honey
11 - raccoon.rh
12 - raccoon.rh.honey

(c) Tom is correct, The issue lies in the constructors of the "V2Radiator" and V3Radiator classes

Tom found the problem in the code, The 'V3Radiator' Constructor calls the 'V2Radiator' Constructor, which adds 5 extra 'SimUnit' objects of type 'V2Radiator' to the list. This means the ratio of retention bots to radiator bots is not 4:3 as expected but rather 4:5, causing high power consumption The total power use would be 120, not 100.

Exercise 10

(A) Be the Compiler.

```
class StaticSuper
{
    static {
            S.o.ut ("supper static block");
           }
    StaticSuper ()
    {
      S.o.ut ("super constructor");
    }
}

Public class StaticTest extends StaticSuper
 {
    Static int rand;
    Static
    {
      rand = (int) (Math.random() * 6);
       S.o.ut ("static block " + rand);
    }

    StaticTest ()
    {
      S.o.ut ("constructor");
    }
```

```
Public static void main (String [] args)
{
    S.o.println("In main");
    StaticTests st = new StaticTests();
}
}
```

=> The 'StaticSuper' class is loaded, and its static block is executed, printing "super static block".

- The StaticTests class is loaded & its static block is executed, printing "static block" followed by a random no. b/w 0 & 5.

- The "main" method is executed, printing "In main".

- The "StaticTests" constructor is called printing "constructor".

(B) True / False

(1) To use False
(2) False
(3) True
(4) False
(5) True
(6) False
(7) True
(8) True
(9) False
(10) False
(11) False
(12) True
(13) False
(14) False

Exercise - 4

Q) Fill in the blanks

When following compilable statements

Collections.sort(myArrayList);

1) what must the class of the objects stored in myArrayList implement?
Comparable

2) what method must the class of the objects stored in myArrayList implement
CompareTo()

3) Can the class of the objects stored in myArrayList implement both Comparator And Comparable?
Yes

Collections.Sort(myArrayList, myCompare);

4) Can the class of the objects stored in myArrayList implement Comparable?
Yes

5) Can the class of objects stored in myArrayList implement comparator?

Yes

6) Must the class of the objects stored in myArrayList implement Comparable?

No

7) Must the class of the objects stored in myArrayList implement Comparator?

No

8) What must the class of the myCompare object implement

Comparator

9) What method must the class of the myCompare object implement

compare ()

13. Reverse Engineer:

```
import java.util.*;

Public class SortMountains
{
    Public static void main (String[] args)
    {
        new SortMountains().go();
    }

    Public void go()
    {
        List<Mountain> mountains =
            new ArrayList<>();
        mountains.add(new Mountain("Longs", 14255));
        mountains.add(new Mountain("Elbert", 14433));
        mountains.add(new Mountain("Maroon", 14156));
        mountains.add(new Mountain("Castle", 14265));
        System.out.println("as entered:\n" + mountains);

        mountains.sort((mount1, mount2) -> mount1.name.
            compareTo(mount2.name));
        S.o.ut("by name:\n" + mountains);

        mountains.sort((mount1, mount2) -> mount2.height -
            mount1.height);
        S.o.ut("by height:\n" + mountains);

    }
}
```

```
Class Mountain
{
    String name;
    int height;

    Mountain(String name, int height)
    {
        this.name = name;
        this.height = height;
    }

    Public String toString()
    {
        return name + " " + height;
    }
}
```

(4) TreeSet exercise

```
import java.util.*;
Public class TreeSet
{
    - - - - -
}
```

(I) what is the result when you compile this code?

It compiles correctly.

(2) If it compiles what is the result when you run do TestTree?

→ It throws an exception

(3) If there is a problem (either compile-time or runtime) with that code, how soon would you find it.

Make a bobble implement Comparable for pass the FirstTrack

Exercise - 12

A) MRed manager

B) for (int i = 1; i <= nums.length; i++)
output += nums.get(i) + " ";

2 3 4 5

C) for (Integer num : nums)
output += nums + " ";

[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]

D) for (int i = 0; i <= nums.length; i++)
output += nums.get(i) + " ";

Compile error

E) for (int i = 0; i <= nums.length; i++)
output += nums.get(i) + " ";

Exception error

F) Code Magnets

```
import java.util.*;
import java.util.stream.*;

Public class CoffeeOrder
{
  Public static void main(String[] args)
  {
    List<String> coffee = List.of("Cappuccino",
    "Americano", "Espresso", "Cortado", "Mocha",
    "Cappuccino", "Flat white", "Latte");
```

```
List<String> coffeeEndingIno = coffees.stream();
        .filter(s -> s.endsWith("o"))
        .sorted()
        .distinct()
        .collect(Collectors.toList());

S.o.ut(coffeesEndingIno);
```

(c) Pool Puzzle.

```
Public class StreamPuzzle
{
    Public class StreamPuzzle
    {
    Public static void main(String args[])
    {
        SongSearch songsearch = new SongSearch();

        songsearch.printTopFiveSongs();
        songsearch.search("The Beatles");
        songsearch.search("The Beach Boys");
    }
}

class SongSearch
{
    Private final List<Song> songs =
        new JukeboxData.Songs().getSongs();
        .sorted(Comparator.comparingInt(Song::getTimesPlayed))
        .map(Song -> Song.getTitle())
```

```
        .limit(5)
        .collect(Collectors.toList());

    System.out.println(topFive);
}

void search(String artist)
{
    Optional<Song> result = songs.stream()
            .filter(song -> song.getArtist().equals(artist))
            .findFirst();
    if(result.isPresent())
    {
        System.out.println(result.get().getTitle());
    }
    else
    {
        System.out.println("No songs found by: " + artist);
    }
}
}
```

Exercise 13.14

(A) Who Am I ?

- I got the whole bowl in my hands
→ JFrame

- Every event type has one of these
→ Listener Interface

- The listeners lazy method
→ actionPerformed()

- This method gives JFrame its size
→ setSize()

- You add code to this method but never call it
→ PaintComponent()

- when the user actually does something, it's an
→ event

- Most of these are event sources
→ Swing Components

- I carry data back to the listener
→ event object

- An addXxxListener() method says an object is an
→ event source

• How a listener sign up
→ addXmListener()

• The method where all the graphics codes.
→ paintComponent;

• I'm typically bound to an instance.
→ inner class.

• The 'g' in (Graphics g) is way of this class
→ Graphics 2D

• The method that gets paintComponent() rolling.
→ repaint().

• The package where most of the Swing are
→ Javax.Swing

(8) Be the Compiler

import javax.Swing.*;
import java.awt.*;
import java.awt.event;

class InnerButton
{
    Private JButton button;

    Public static void main(String[] args)
    {
        InnerButton gui = new InnerButton();

```
        gui.go();
    }

    public void go()
    {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        button = new JButton("A");
        button.addActionListener(new ButtonListener());

        frame.getContentPane().add
        {
            BorderLayout.SOUTH, button);

        frame.setSize(250, 100);
        frame.setVisible(true);
    }

    Class ButtonListener Implements ActionListener
    {
        public void actionPerformed(ActionEvent e)
        {
            if(button.getText().equals("A"))
            {
                button.setText("B");
            }
            else {
                button.setText("A");
            }
        }
    }
```

(c) Pool Puzzle

```
import javax.swing.*;
import java.awt.*;
import java.util.concurrent.TimeUnit;

Public class Animate
{
    int x = 1;
    int j = 1;
    Public static void main (String [] args)
    {
        Animate gui = new new Animate ();
        gui.go();
    }
}

Public void go ()
{
    JFrame frame = new JFrame ();
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

    MyDrawP drawP = newDrawP ()
    frame.getContentPane ().add (drawP);
    frame.setSize (500, 270);
    frame.setVisible (true);
    for (int i=0 ; i<124 ; i++, j++, x++)
    {
        x++;
        drawP.Repaint ();
    try
    {
        TimeUnit.MILLISECONDS.sleep (50);
    }
```

```
catch (Exception ex)
{
}
}
}

class MyDrawPanel
{
    public void paintComponent(Graphics g)
    {
        g.setColor(Color.white);
        g.fillRect(0, 0, 500, 250);
        g.setColor(Color.blue);
        g.fillRect(x, y, 500-x*2, 250-y*2);
    }
}
}
```

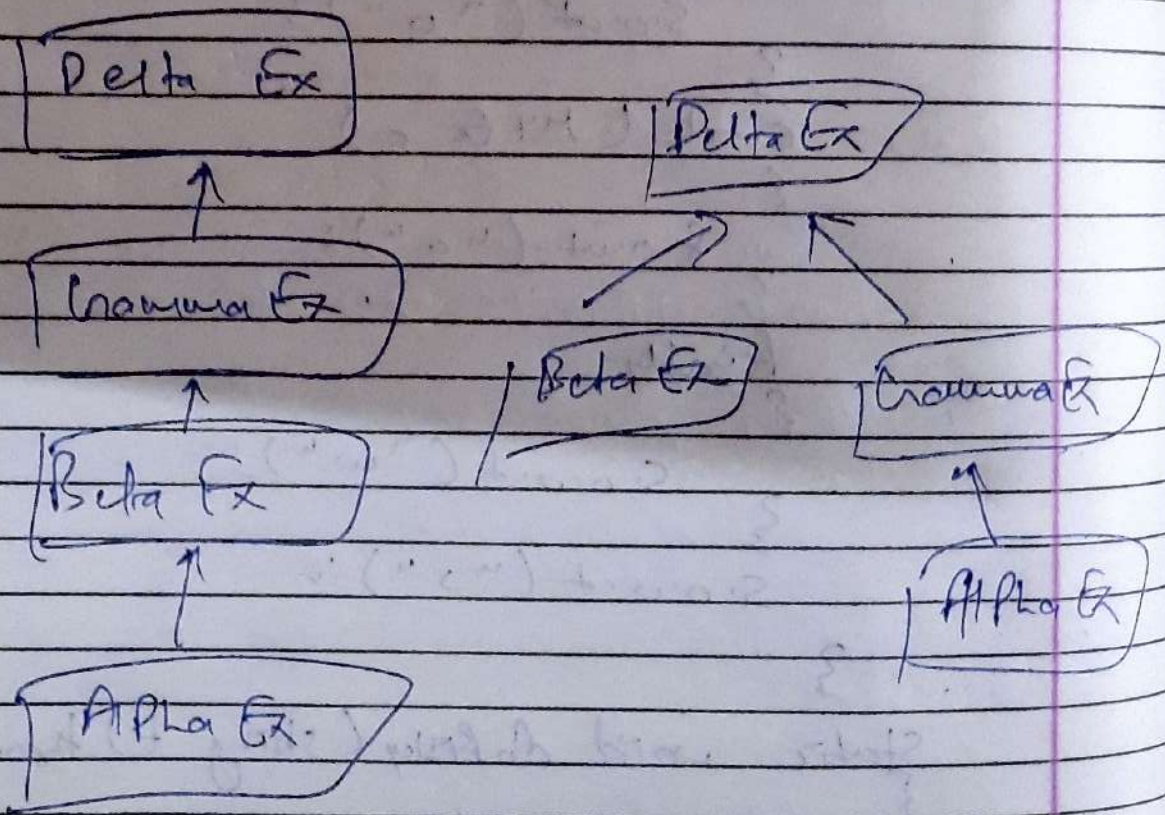
Exercise - 13

(A) Core Magnet

```
class MyEx extends Exception
{ }
public class ExTestPrac
{
    Public static void main (String [] args)
    {
        String test = arg [0];
    try {
        S.o.ut ("t");
        doRisky (test);
        S.o.ut ("o");
    }
    catch (MyEx e)
    {
        S.o.ut ("a");
    }
    finally ...
    {
        S.o.u.t ("w");
    }
    S.o.u.t (" ");
    }
Static void doRisky (String t) throws MyEx
{
    Static S.o.ut ("h");
```

```
if ("Yes".equals(t))
{
    throw new MyEx();
}
System.out.println("Yes");
}
}
```

(B) Sharpen your pencil



Exercise - 15

(A) which code goes which layout

(1) → (C)

(2) → (D)

(3) → (E)

(4) → (A)

(5) → (B)

Exercise - 16

True / False

(1) False
(2) False
(3) True
(4) False
(5) True
(6) False
(7) False
(8) false
(9) True
(10) True

(11) True
(12) True
(13) false
(14) false
(15) True
(16) True
(17) false.

Code Implied

```
import java.io.*;

class PurgeonCourse implements Serializable
{
    public int x = 2;
    transient long y = 9;
    private short z = 5;

    int getx()
    {
        return x;
    }
    long gety()
    {
        return y;
    }
    short getz()
    {
        return z;
    }
}

class PurgeonTest
{
    public static void main(String[] args)
    {
        PurgeonCourse d = new PurgeonCourse();
        System.out.println(d.getx() + d.gety() + d.getz());
    }
}
```

```
try {
    FileOutputStream fos = new FileOutputStream
                           ("dg.ser");
    ObjectInputStream ois = new ObjectInputStream(fis);
    d = (DungeonGame) ois.readObject();
    ois.close();
}

catch (Exception e)
{
    e.printStackTrace();
}

S.O.ut(d.getX() + d.getY() + d.getZ());

}
}
```