Niharika Mall

**Chapter 1 :**

A – Wouldn’t compile

while( x<10)

{  
x = x+1; // to prevent infinite loop

if(x>3)

….

}

B – wouldn’t compile

class B // need to declare class

{

}

C – wouldn’t compile

Class Exercise1c

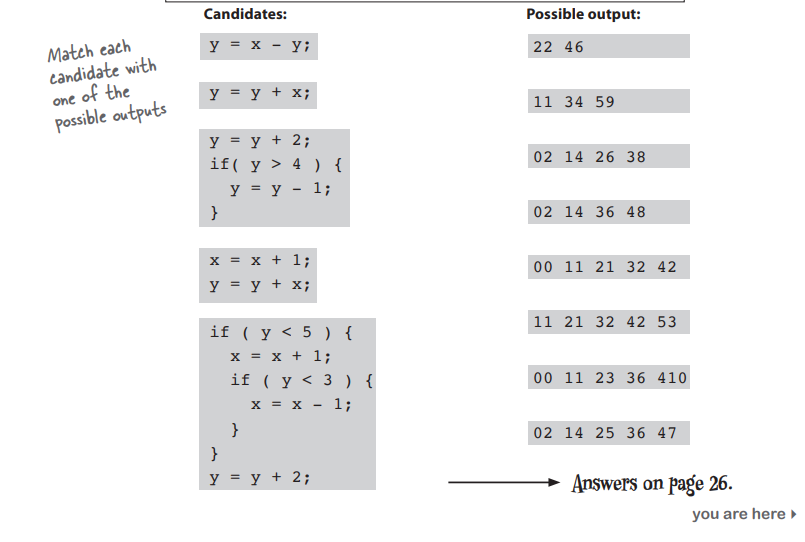
{  
public static void main ( String args []) // need to create main method

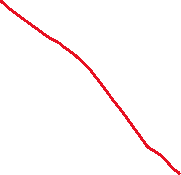
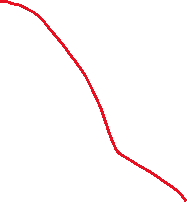
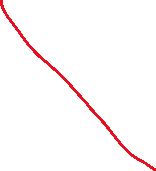
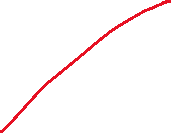
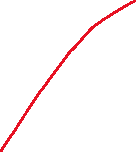
{

}

Mixed Messages







Pool Puzzle

While ( x<4)

{  
System.out.print(“a”);

If( x<1)

{  
System.out.print(“ “);

}

System.out.print(“n”);

If( x>1)

{  
System.out.print(“oyster “);

X=x+2;

}

If( x==1)

{  
System.out.print(“ noys“);

}

If( x<1)

{  
System.out.print(“oise “);

}

System.out.println();

X = x+1;

}

}  
}

**Chapter 2 :**

A – wouldn’t compile

Class StreamingSongTestDrive()

{

Public static void main ( String args [])

{  
StreamingSong song = new StreamingSong(); // need to declare object of class

…

}

}

B – wouldn’t compile

Void play()

{  
System.out.println(“Playing episode “, episodeNumber) ; // needs a play function

}

Code Magnets

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 = false;

d.playTopHat();

if (d.snare == true)

{ d.playSnare();

}

}

}

Who am I

Class

Object

Class

Object

Class

Instance Variable

Object

Instance Variable

Object

Class

Object

Class

Instance Variable

**Chapter 3 :**

A – wouldn’t compile

public static void main(String[] args)

{

Books[] myBooks = new Books[3];

int x = 0;

myBooks[0] = new Books(); // need to declare each object individually

myBooks[1] = new Books();

myBooks[2] = new Books();

…

}

B – wouldn’t compile

While ( z<3) // size of array is 3 so 0,1,2 so indexing needs to be less than 3

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 ref;

while (y < 4) {

ref = index[y];

System.out.print("island = ");

System.out.println(islands[ref]); y = y + 1;

}

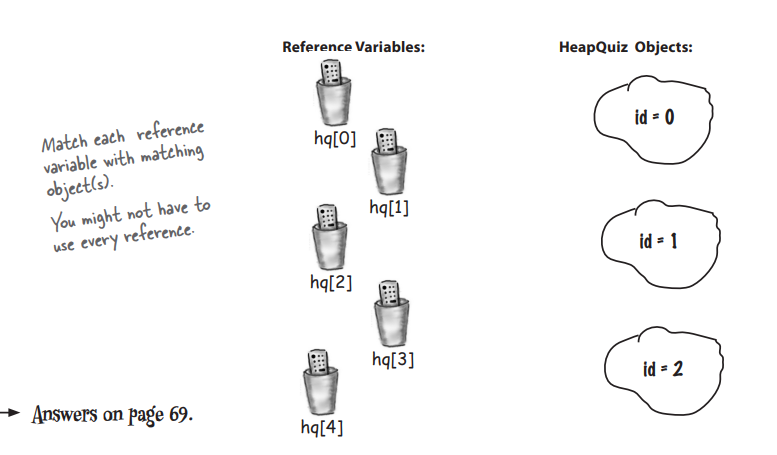
}

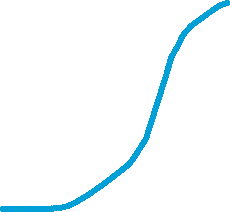
}

The case of pilfered references

We cannot access the first 9 elements of Kates object because they get updates in every iteration. Thats why Tawny chose Bobs method instea

A Heap Of Trouble





**Chapter 4:**

A – compiles

B – wouldn’t compile

String getTime() // return type is String

{

…

return time  
}

Who am I

Getter, setter, methods

Return

Argument

Encapsulation

Pass by value

Instance variables

Argument

Getter

Public

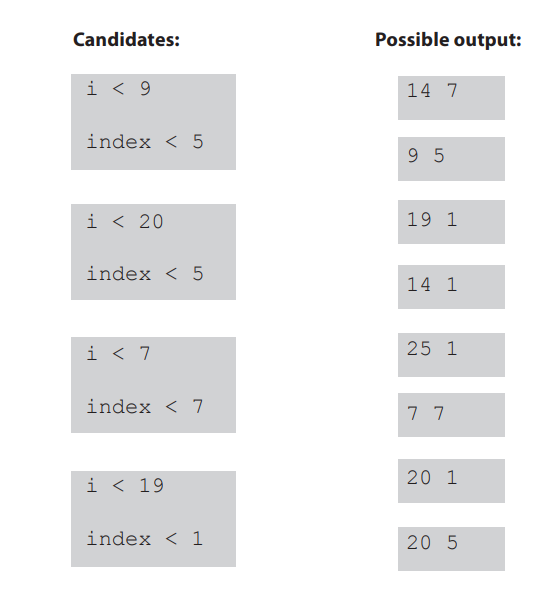
Method

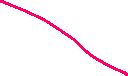
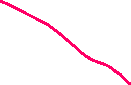
Setter

Getter , setter

Return

Mixed Messages





Five Minute Mystery

Buchan said he made his WareHousing methods private but didn’t mention the instance variables. He should mark those as private too.

**Chapter 5:**

Be the JVM

Output would be

13 15 x = 6

Code Magnets

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++;

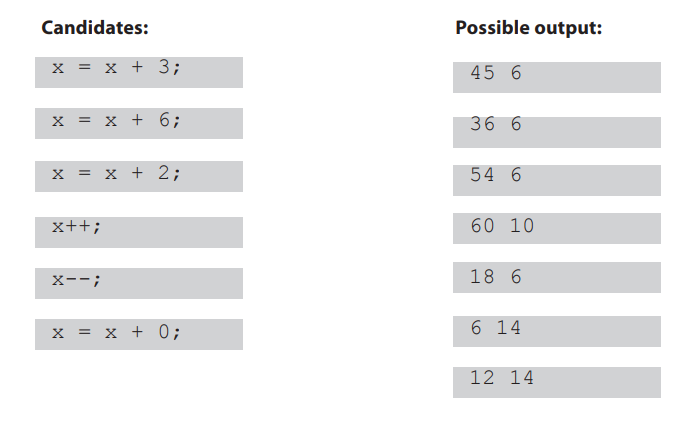
}

}

}

}

Mixed Messages





**Chapter 6 :**

**Code Magnets**

import java.util.ArrayList;

public class ArrayListMagnet

{

public static void main(String[] args)

{

ArrayList a = new ArrayList();

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 list)

{

for (String element : list)

{

System.out.print(element + " ");

}

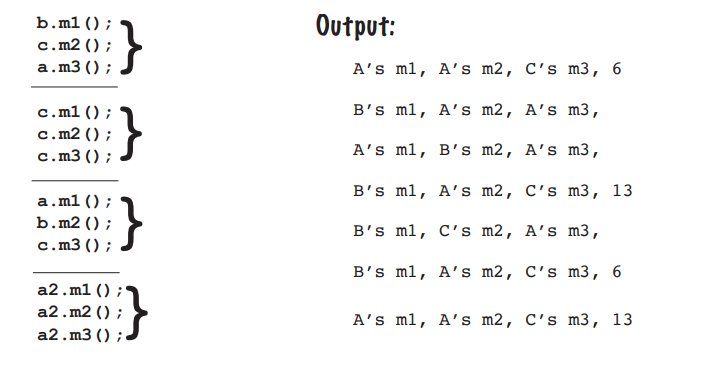
System.out.println();

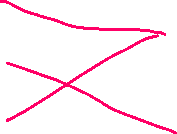
}

}

**Chapter 7:**

Mixed Messages





Be the compiler

1 – will comile

2 – will not

3 – will compile

4 - will compile

**Chapter 8:**

Whats the declaration

2.

Public abstract class Top{}

Public class Tip extends Top {}

3.

Public abstract class Fee{}

Public abstract class Fi extends Fee {}

4.

Public interface Foo{}

Public class Bar implements Foo{}

Public class Baz extends Bar{}

5.

Public interface Zeta{}

Public class Alpha implements Zeta{}

Public interface Beta{}

Public class Delta extends Alpha implements Beta {}

Pool Puzzle

Interface Noise {  
public int Method();

}

Abstract class Picasso implements Nose {

Public int Method();

Return 7;

}  
}

Class Clowns extends Picasso

{

}

Class Acts extends Picasso

{

Public int method()

{  
return 5;

}  
}

**Chapter 9:**

Be the garbage collecter

Only 2 and 8 will cause an additional object to be eligible

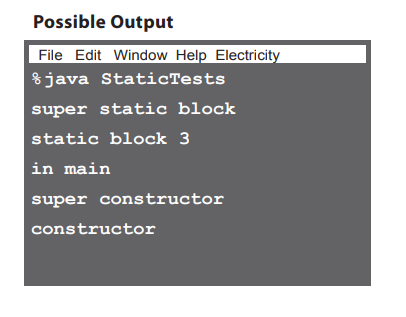
5 minute mystery

Tom suspects that it was using more power because it was passing an array list and hence 5 extra simunits.

**Chapter 10:**

Be the compiler

It would compile and this is the output



True or False

1. 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