Q1. What is the output of the following program? [Topic03 P7]

```
class X {
    private int xValue;
    private X xChild;

    public X(int xValue, X xChild) {
        this.xValue = xValue;
        this.xChild = xChild;
    }

    public String toString() {
        return xChild + " " + xValue;
    }
}

public class Tester{
   public static void main(String[] args) {

       X obj = new X(99, new X(88, new X(77, null)));
       System.out.println(obj);
   }
}
```

Output:

Q2. [Topic03 - Page 10]

```
class X {
    private int data;
    public X(int d) {data=d*2;}
    public String toString() {return ""+data;} //a trick to make a string quickly

    public void doSomething(X r) {
        X s = new X(8);
        System.out.println(this.data);
        System.out.println(r.data);
        System.out.println(s.data);
    }
}

public class Main_X_3Objs {
   public static void main(String[] args) {
       X a = new X(1);
       X b = new X(15);
       a.doSomething(b);
   }
}
```

(i) Which one is the output of the program?

Output:
```
2
30
16
```

Output:
```
16
16
16
```

Output:
```
2
2
2
```

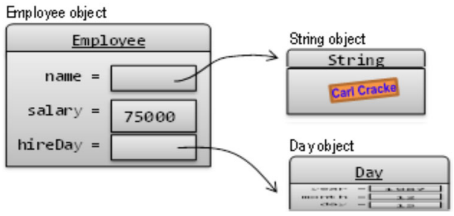
(ii) Find the following from the **`doSomething`** method:
(a) Implicit parameter (b) Explicit parameter (c) Local variable

(iii) The **`doSomething`** method of class X can / cannot access private member (data) defined in class X, even if the members do not belong to the implicit parameter ↙Calling object, 'this'

Q3. [Topic03 - Page 11]

Line 2 of the following changes the name of an employee by calling a `setName()` method.
Can it change the content inside the name String object? Answer: yes / no.
By drawing, illustrate what line 2 can do.

```
/* line 1*/ Employee e = new Employee(
                "Carl Cracker", 75000,
                1987, 12, 15);
/* line 2*/ e.setName("Helena");
```



Q4 [Topic03 P13]

What will happen when the following program runs?
Explain why the commented statements in line X1, X2 are wrong.

```
class A {
    public int x1;
    public static int x2;

    public void m1() {

        m2();
    }

    public static void m2() {

        //x1+=100; //<== this is invalid. [line X1]
        x2+=100;
        //m1(); //<== invalid [line X2]
    }
}

public static void main(String[] args) {

    (new A()).m1();
    (new A()).m1();
}
```

Q5. [Topic03 P16]

The `RandomNumber` class below is designed to give random numbers. Sample usage:

`new RandomNumber()` - give a number in 0 .. 99

`new RandomNumber(n)` - give a number 0 .. n-1

However it has a compile-time error: *The constructor RandomNumber() is undefined*

Your tasks: Explain why there exists such an error and give a proper solution.

```
public class RandomNumber {
    private int r=(int)(Math.random()*100); //0..99
    public RandomNumber(int n) { r=(int)(Math.random()*n); }
    public String toString() {return ""+r;}
    public static void main(String[] args) {
        System.out.println(new RandomNumber(1000)); //Expected: 0..999 [OK]
        System.out.println(new RandomNumber()); //Expected: 0..99 [ERROR!]
    }
}
```

* Since a constructor is written by ourselves, JAVA does not generate the no-argument constructor for us.

Q6. [Review of Basic Casting]

On the right is a java program.

The output from lines 3 and 4 are shown in the comments.

```
        public static void main(String [] args)
        {
Line 1      int i=97; char c;
Line 2      c=(char)i;
Line 3      System.out.println(i); //output: 97
Line 4      System.out.println(c); //output: a
        }
```

(i) Does the data type of `i` change after line 2? Yes / No

(ii) Which saying(s) is/are accurate concerning `(char)i`?

- ☐ (a) `(char)i` means to convert i to char.
- ☐ (b) `(char)i` means to convert the value of i to a char value.
- ☐ (c) `(char)i` means to provide a char value by converting from the value of i.
- ☐ (d) `(char)i` can be used if we want to consider the value of i as a char.

Q7. Inheritance and **Casting** [Topic04 Page 8]

(a) What is the output from this program?

```
class A {
    public String f1() { return "A solution";}
}

class B extends A {
    public String f1() { return "Best solution";}
}

public class Main {
    public static void main(String [] args) {
        go(new B());
    }

    public static void go(A x) {
        String result1 = ((B)x).f1(); //"A solution" or "Best solution"?
        String result2 = x.f1();      //"A solution" or "Best solution"?
        System.out.println(result1 + ", " + result2);
    }
}
```

(b) What's wrong with the following sayings?

(i) Given a variable declared as `A x;` (where A is a class; which has a subclass, class B)

* `(B)x` means to change the type of x to class B.

* If x actually refers to a class B object and both class A and class B have a non-static method of the same name, `f1`, then due to dynamic binding, `x.f1()` will run the method defined in class B.

(ii) Dynamic Binding – The compiler automatically selects the appropriate non-static method at runtime

Q8 [Topic04 Page 8]

Which of the following statements, if inserted into the program below, will output 200?
Use any 2 statements in the program, explain what is **up-casting**, what is **down-casting**.

```
class A {
    private int x;
    public A() {x=100;}
    public void m1() { System.out.println(x);}
}

class B extends A {
    private int y;
    public B() {y=200;}
    public void m1() {System.out.println(y);}
}

public static void main(String[] args)
{
    A i = new A();
    B j = new B();


}
```

```
i.m1();
j.m1();
((A)j).m1();
((B)i).m1();
```

Q9 [Topic04 Page 11 and 13]

Read the code below. Which line(s) contain invalid code? Why?

```
Integer i=3;    //line 1
String s;       //line 2
s = (String)i;  //line 3
Object o;       //line 4
o = i;          //line 5
i = (Integer)o;//line 6
```

* String and Integer are not pair of super_class / sub_class.

Q10. (i) What is the output of the following program? [Topic 4 P.13]
(ii) Rewrite the for-loops as for-each loops [Topic 2 P.13]

```
public static void main(String[] args) {

  Integer[] arr = new Integer[3]; // Simply treat Integer as int (See Topic 04)
  arr[0] = 100;  arr[1] = 101;  arr[2] = 109;

  ArrayList<Integer> alist = new ArrayList<Integer>();
  alist.add(100);  alist.add(101);  alist.add(109);

  for (int i = 0; i < arr.length; i++)
     System.out.print(arr[i] + " ");

  System.out.println();

  for (int i = 0; i < alist.size(); i++)
     System.out.print(alist.get(i) + " ");

}
```

Output:

Rewrite as for-each loops:

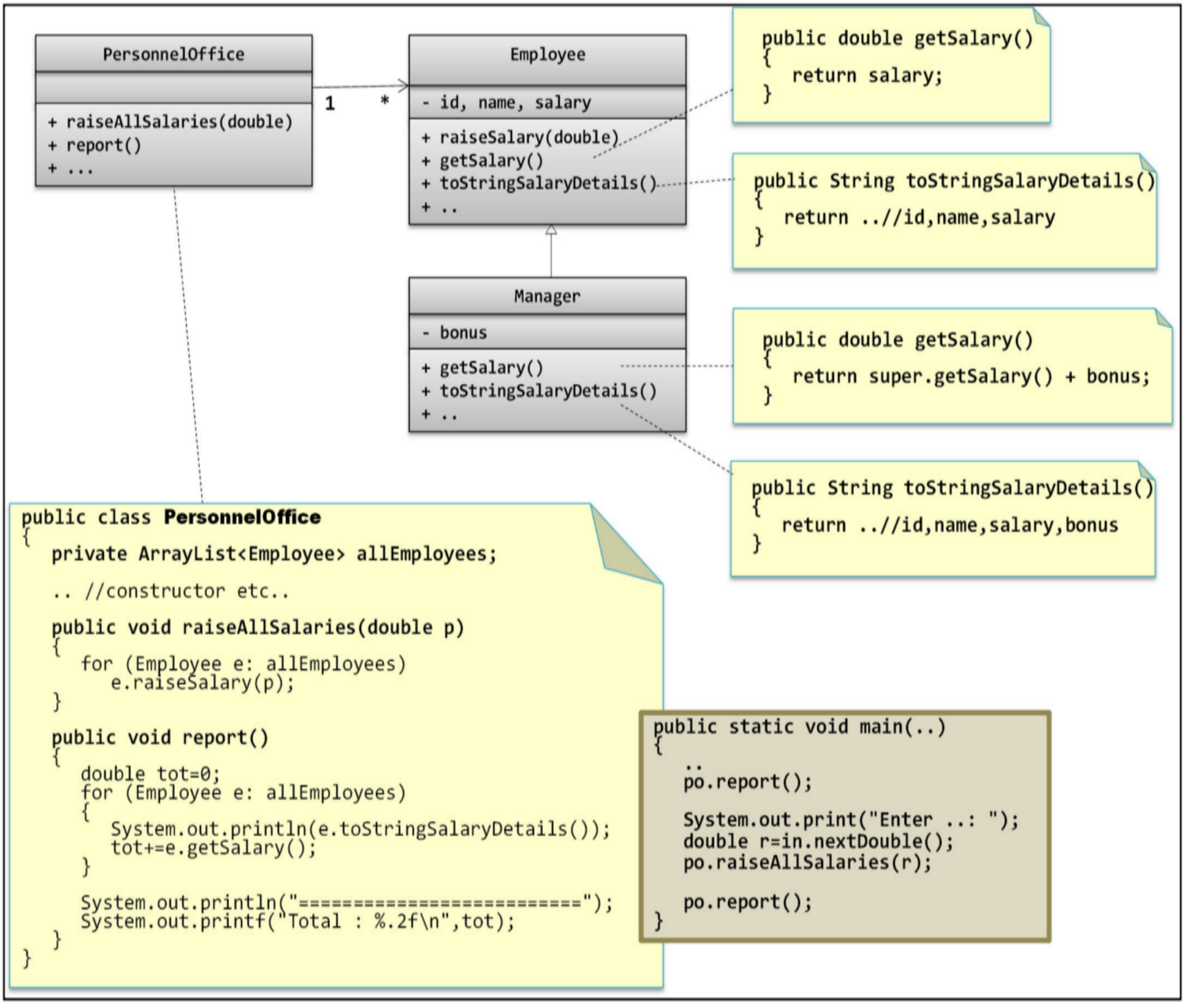
```
for(                        )
  System.out.print(     + " ");

for(                        )
  System.out.print(     + " ");
```

(iii) In the program, we add the following code. Complete the blanks below to show program outputs.

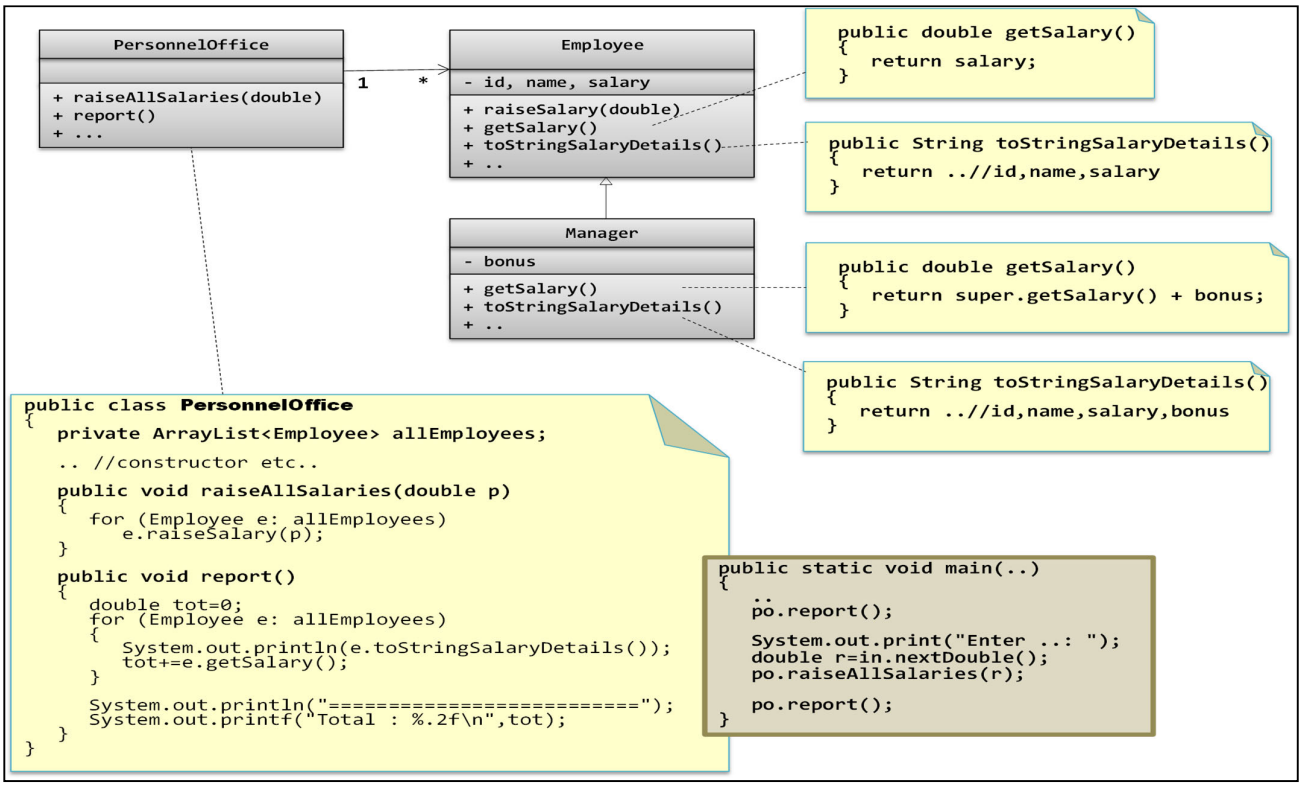
```
// printing using toString
System.out.println(Arrays.toString(arr));   //output: [100, 101, 109]
System.out.println(alist.toString());       //output: [100, 101, 109]
System.out.println(); System.out.println();

// Try to change element values
for (Integer e : arr)
  e++;

for (Integer e : alist)
  e++;

System.out.println(Arrays.toString(arr));   //output:____________________
System.out.println(alist.toString());       //output:____________________
System.out.println(); System.out.println();

// Try to change add or remove element in arraylist
for (Integer e : alist)
  if (e == 109)
    alist.add(110);

for (Integer e : alist)
  if (e == 109)
    alist.remove(e);

System.out.println(Arrays.toString(arr));   //output:____________________
System.out.println(alist.toString());       //output:____________________
```

Note:
In a *for-each loop*, java performs iteration using an *iterator* (will teach in Topic08). The collection should not be modified.
That is, we should not insert or remove elements from the collection. Otherwise, we might get an exception.

```
Exception in thread "main" java.util.ConcurrentModificationException
        at java.util.ArrayList$Itr.checkForComodification(Unknown Source)
        at java.util.ArrayList$Itr.next(Unknown Source)
    at Main.main(Main.java:57)
```


PersonnelOffice
+ raiseAllSalaries(double)
+ report()
+ ...
1
*
Employee
- id, name, salary
+ raiseSalary(double)
+ getSalary()
+ toStringSalaryDetails()
+ ..
Manager
- bonus
+ getSalary()
+ toStringSalaryDetails()
+ ..
public double getSalary()
{
return salary;
}
public String toStringSalaryDetails()
{
return ..//id,name,salary
}
public double getSalary()
{
return super.getSalary() + bonus;
}
public String toStringSalaryDetails()
{
return ..//id,name,salary,bonus
}
public class PersonnelOffice
{
private ArrayList<Employee> allEmployees;
.. //constructor etc..
public void raiseAllSalaries(double p)
{
for (Employee e: allEmployees)
e.raiseSalary(p);
}
public void report()
{
double tot=0;
for (Employee e: allEmployees)
{
System.out.println(e.toStringSalaryDetails());
tot+=e.getSalary();
}
System.out.println("==========");
System.out.printf("Total : %.2f\n",tot);
}
}
public static void main(..)
{
..
po.report();
System.out.print("Enter ..: ");
double r=in.nextDouble();
po.raiseAllSalaries(r);
po.report();
}

Q11. (a) Explain: "reuse" and "redefine" found in Lab05_Q4 program:



Answer:

(1) Reuse: The _____ method defined in the _____ class is reused in the _____ class.

(2) Redefine: The _____ methods defined in the _____ class are redefined in the _____ class.

(b) Read the `report()` method in `PersonnelOffice.java` and note the statement: `tot+=e.getSalary();`
If some employees e are managers, will the bonus be counted?

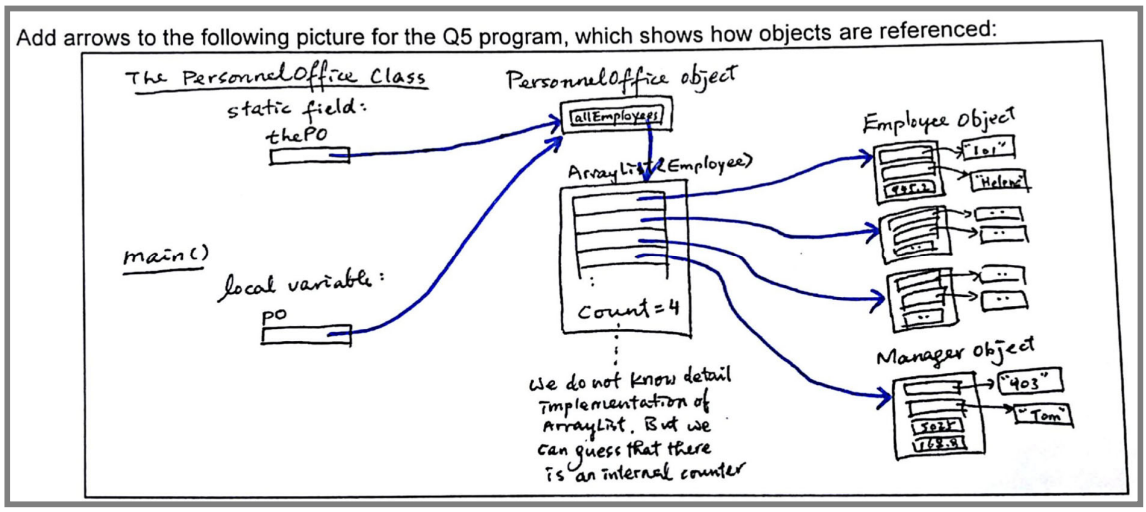
[Ref: **Polymorphism and Dynamic Binding** in Page 7 of Topic 04 Inheritance]

(c) Recall Lab05-Q5:

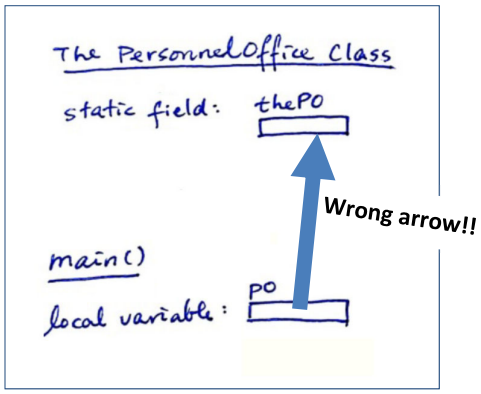
```
public class PersonnelOffice
{
    //Reference to an array list of employee objects
    private ArrayList<Employee> allEmployees;

    //There should be only one personnel office.  So we declare it as a static field in the class.
    private static PersonnelOffice thePO = new PersonnelOffice();

    //Return the personnel office
    public static PersonnelOffice getInstance() {return thePO;}

    //Constructor
    private PersonnelOffice()
    {
        allEmployees = new ArrayList<Employee>();
    }

    ..
```

```
public static void main(String[] args) ...
{
    ..

    PersonnelOffice po = PersonnelOffice.getInstance();
    ..
```

The following is the model answer of Lab05 Progress exercise:



(i) The following arrow is wrong. Why?



(ii) The numbers are inside object fields but strings are not. Why?