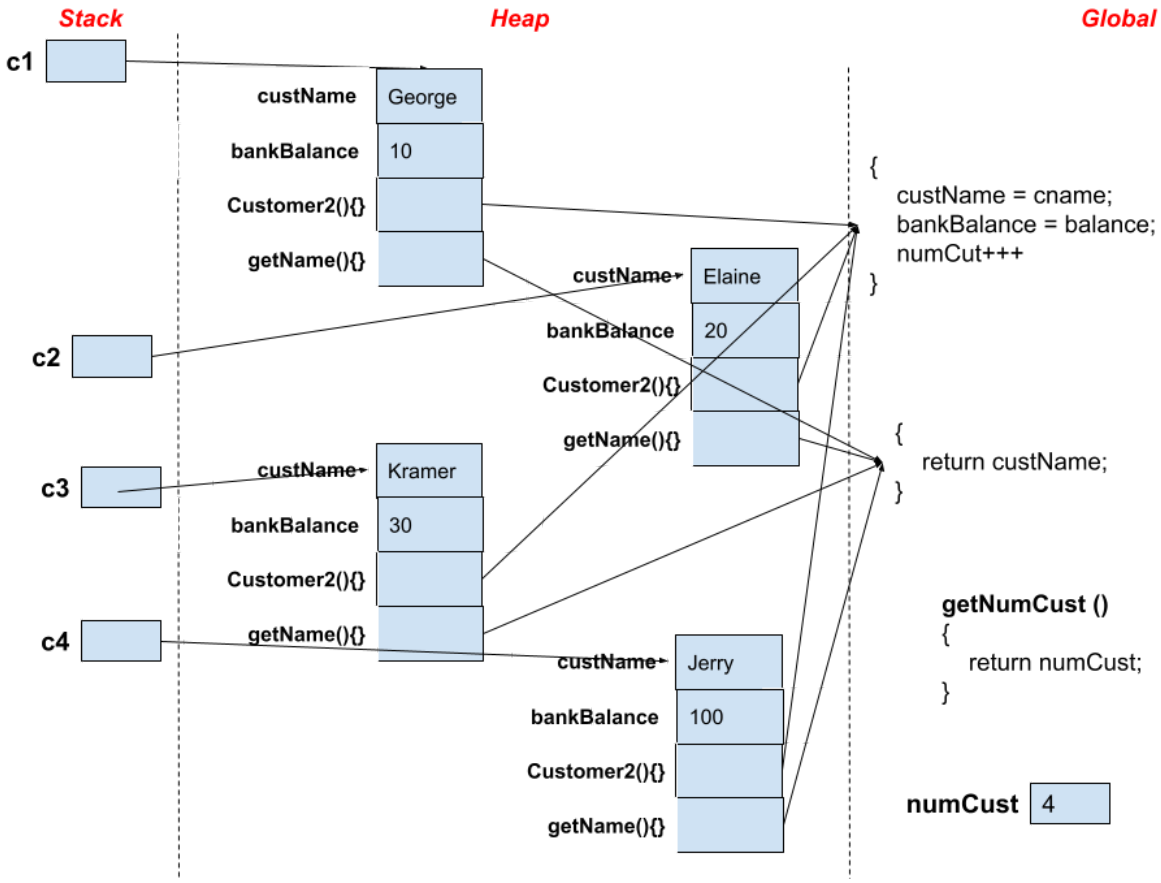
Dmitri Stanchevici
Unit 6
Module 4

Ex. 4.2



Ex. 4.4

The error in Customer3 is

```
Customer3.java:8: error: non-static variable custName cannot be referenced from a
static context
    custName = "Blah";
    ^
1 error
```

custName is an **instance variable**, and as such it cannot be directly accessed by a static method test(), which belongs not to an instance, but to the whole class. To access custName, test() would have to create an instance of Customer3 with the operator new and then access this instance's custName variable through the dot operator.

Ex. 4.5

Printed is

```
Person: Name=George, ssn=111-11-1234
```

Ex. 4.6

Printed is

```
Person: Name=Elaine, ssn=333-33-4567, age=35
```

Ex. 4.7

Printed is

```
PersonV3: name=Kramer, ssn=666-66-1234, age=38
Kramer, 666-66-1234
```

When `System.out.println (p);` is executed, **toString()** in **Person** (the superclass) is called.

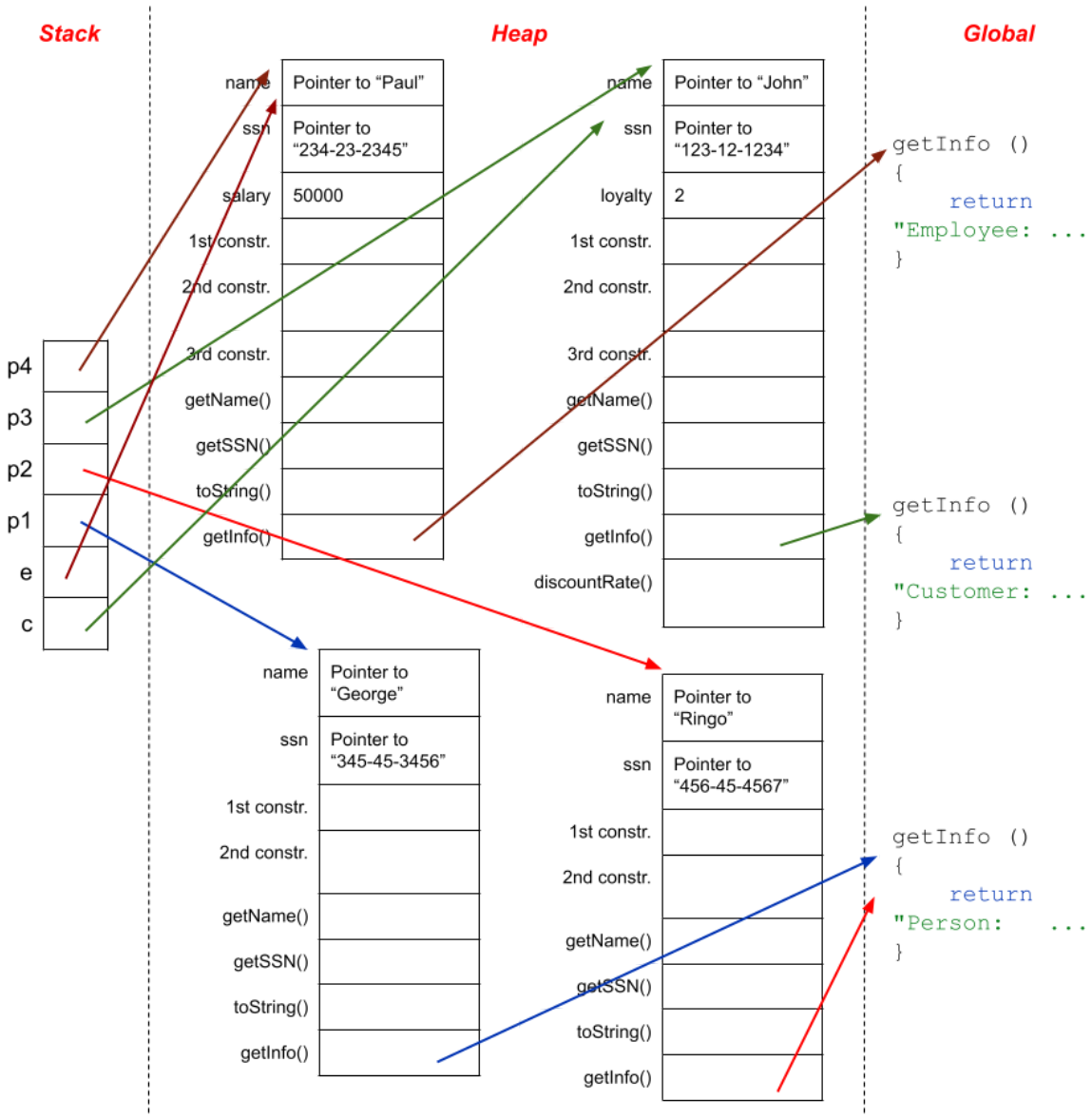
Ex. 4.8

Printed is

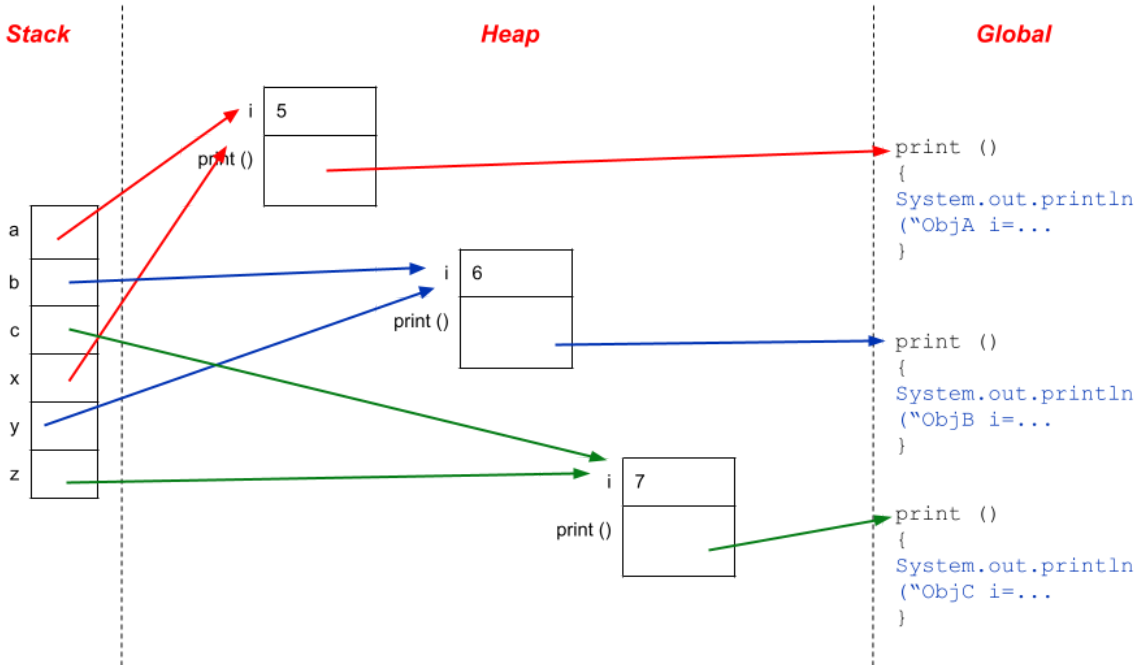
```
Customer: name=John, 2 years with the company
Employee: name=Paul, salary=50000

Person data:
Person: Name=George, ssn=345-45-3456
Person: Name=Ringo, ssn=456=45-4567
Customer: name=John, 2 years with the company
Employee: name=Paul, salary=50000
```

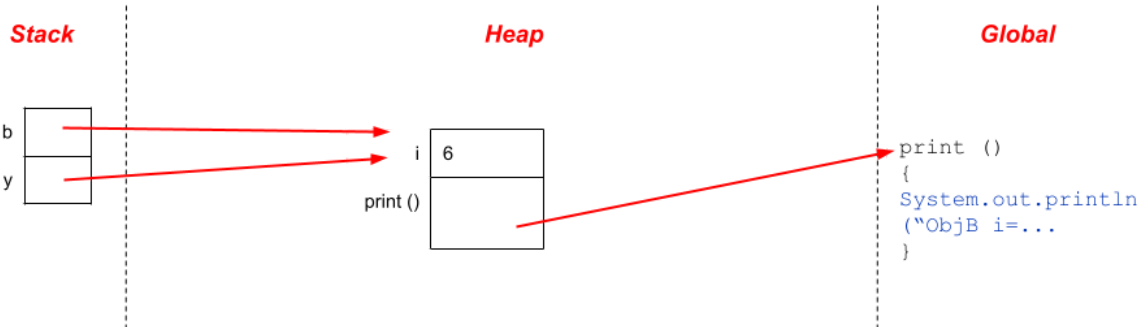
Ex. 4.9


Stack
Heap
Global
p4
p3
p2
p1
e
c
name
Pointer to "Paul"
ssn
Pointer to "234-23-2345"
salary
50000
1st constr.
2nd constr.
3rd constr.
getName()
getSSN()
toString()
getInfo()
name
Pointer to "John"
ssn
Pointer to "123-12-1234"
loyalty
2
1st constr.
2nd constr.
3rd constr.
getName()
getSSN()
toString()
getInfo()
discountRate()
name
Pointer to "George"
ssn
Pointer to "345-45-3456"
1st constr.
2nd constr.
getName()
getSSN()
toString()
getInfo()
name
Pointer to "Ringo"
ssn
Pointer to "456-45-4567"
1st constr.
2nd constr.
getName()
getSSN()
toString()
getInfo()
getInfo () { return "Employee: ... }
getInfo () { return "Customer: ... }
getInfo () { return "Person: ... }

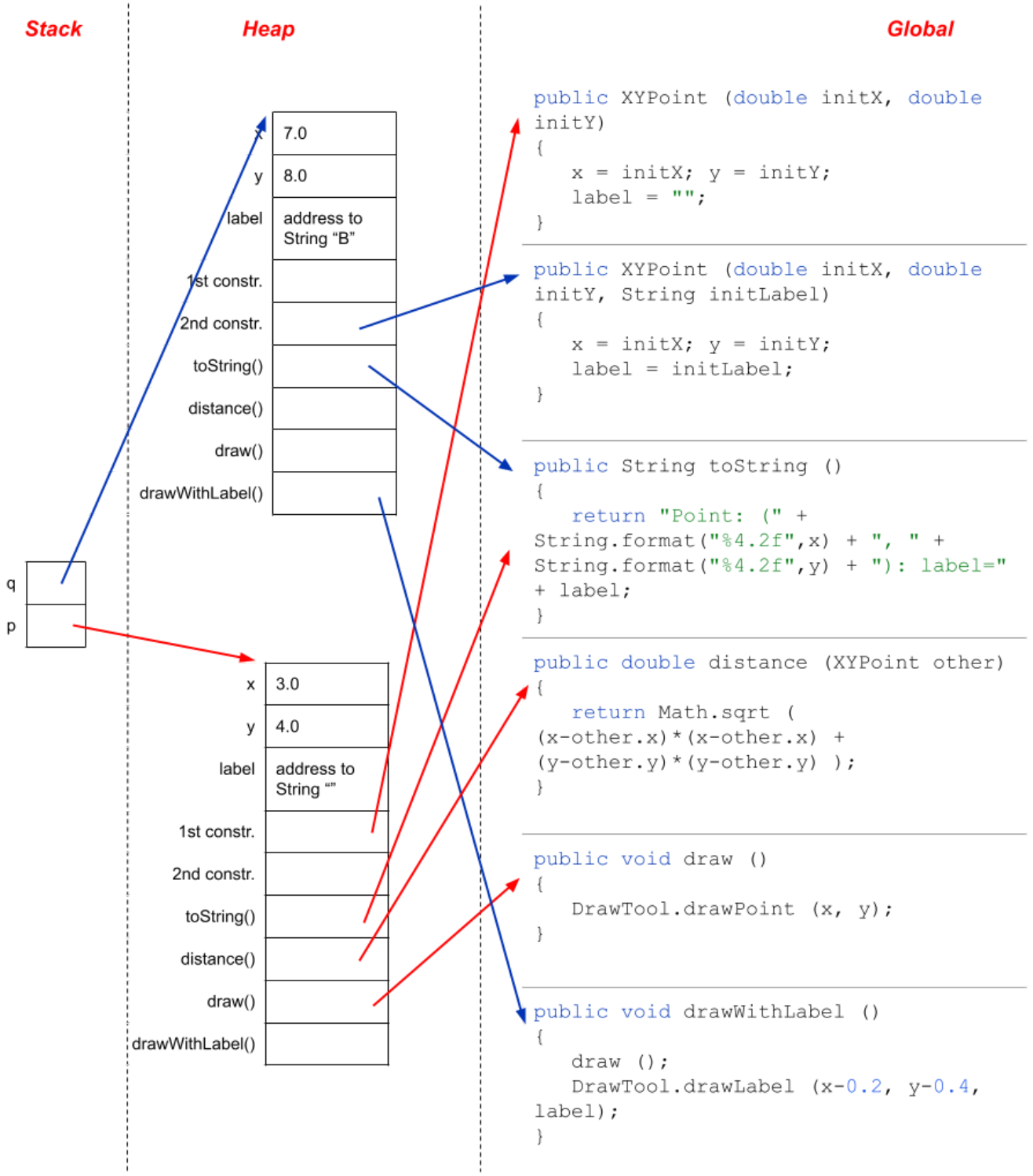
Ex. 4.11


Stack
Heap
Global
i
5
print ()
a
b
c
x
y
z
i
6
print ()
i
7
print ()
print ()
{
System.out.println
("ObjA i=...
}
print ()
{
System.out.println
("ObjB i=...
}
print ()
{
System.out.println
("ObjC i=...
}

Ex. 4.12

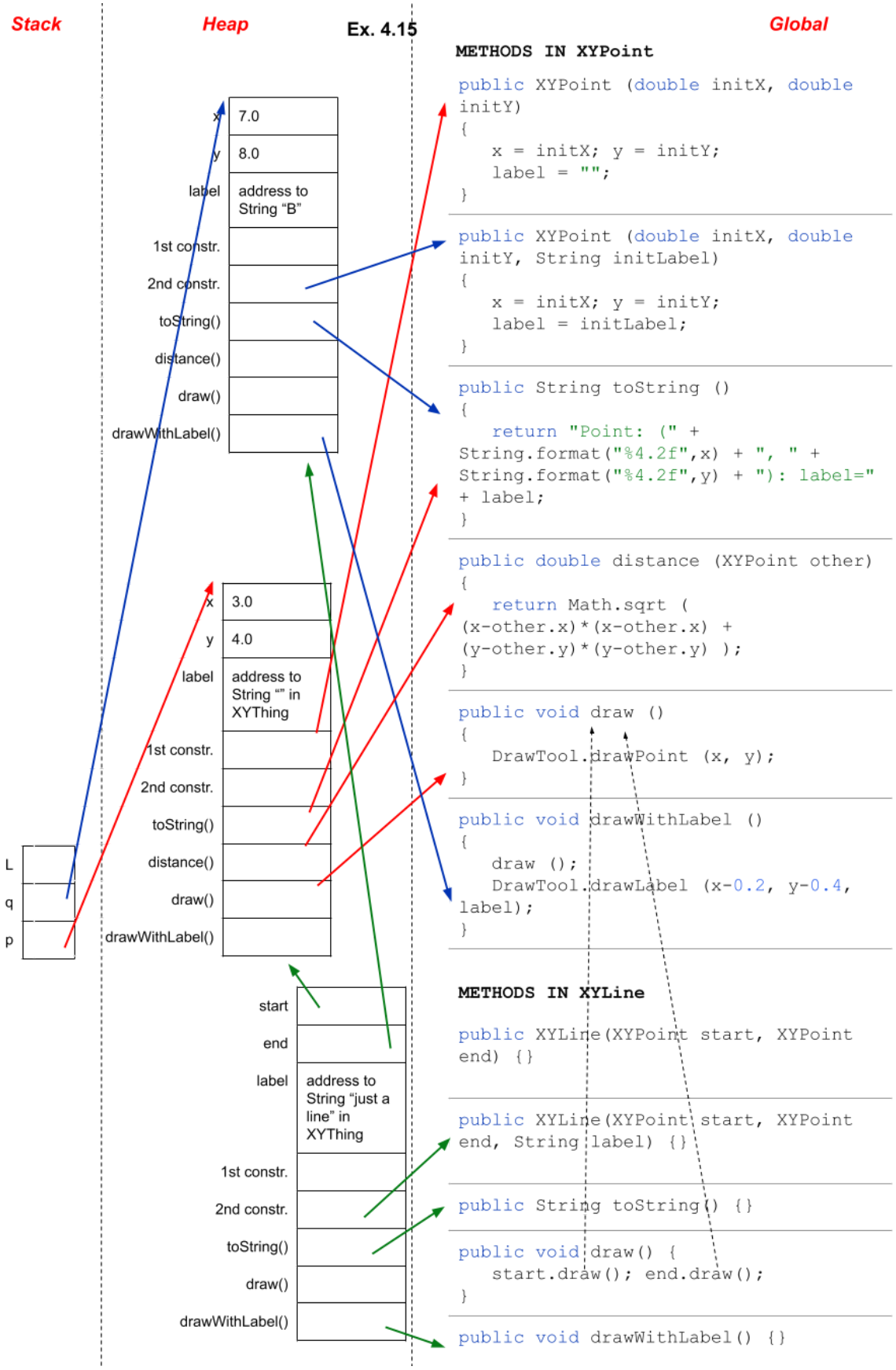

Stack
Heap
Global
b
y
i
6
print ()
print ()
{
System.out.println
("ObjB i=...
}

Ex. 4.14


Stack
Heap
Global
x
7.0
y
8.0
label
address to String "B"
1st constr.
2nd constr.
toString()
distance()
draw()
drawWithLabel()
q
p
x
3.0
y
4.0
label
address to String ""
1st constr.
2nd constr.
toString()
distance()
draw()
drawWithLabel()
public XYPoint (double initX, double initY)
{
x = initX; y = initY;
label = "";
}
public XYPoint (double initX, double initY, String initLabel)
{
x = initX; y = initY;
label = initLabel;
}
public String toString ()
{
return "Point: (" + String.format("%4.2f",x) + ", " + String.format("%4.2f",y) + "): label=" + label;
}
public double distance (XYPoint other)
{
return Math.sqrt ((x-other.x)*(x-other.x) + (y-other.y)*(y-other.y));
}
public void draw ()
{
DrawTool.drawPoint (x, y);
}
public void drawWithLabel ()
{
draw ();
DrawTool.drawLabel (x-0.2, y-0.4, label);
}

Ex. 4.15

See page below.


Stack
Heap
Ex. 4.15
Global
x
7.0
y
8.0
label
address to String "B"
1st constr.
2nd constr.
toString()
distance()
draw()
drawWithLabel()
x
3.0
y
4.0
label
address to String "" in XYThing
1st constr.
2nd constr.
toString()
distance()
draw()
drawWithLabel()
L
q
p
start
end
label
address to String "just a line" in XYThing
1st constr.
2nd constr.
toString()
draw()
drawWithLabel()
METHODS IN XYPoint
public XYPoint (double initX, double initY)
{
x = initX; y = initY;
label = "";
}
public XYPoint (double initX, double initY, String initLabel)
{
x = initX; y = initY;
label = initLabel;
}
public String toString ()
{
return "Point: (" + String.format("%4.2f",x) + ", " + String.format("%4.2f",y) + "): label=" + label;
}
public double distance (XYPoint other)
{
return Math.sqrt ((x-other.x)*(x-other.x) + (y-other.y)*(y-other.y));
}
public void draw ()
{
DrawTool.drawPoint (x, y);
}
public void drawWithLabel ()
{
draw ();
DrawTool.drawLabel (x-0.2, y-0.4, label);
}
METHODS IN XYLine
public XYLine(XYPoint start, XYPoint end) {}
public XYLine(XYPoint start, XYPoint end, String label) {}
public String toString() {}
public void draw() {
start.draw(); end.draw();
}
public void drawWithLabel() {}

Ex. 4.19

In **TestPoly.java:**

Class	Constructors

Person	`public Person (String nameIn, String ssnIn)`
	`{`
	`  name = nameIn; ssn = ssnIn;`
	`}`
	
	`public Person ()`
	`{`
	`  name = ssn = "Not initialized";`
	`}`

Customer extends Person	`public Customer (String nameIn, String ssnIn, int loyaltyIn)`
	`{`
	`  name = nameIn; ssn = ssnIn; loyalty = loyaltyIn;`
	`}`

Employee extends Person	`public Employee (String nameIn, String ssnIn, int salaryIn) {`
	`  name = nameIn;`
	`  ssn = ssnIn;`
	`  salary = salaryIn;`
	`}`

Ex. 4.20

The compiling error is

```
ConstructorExample.java:24: error: constructor ObjB2 in class ObjB2 cannot be applied
to given types;
      ObjB2 b = new ObjB2 (5);
                ^
  required: no arguments
  found:    int
  reason: actual and formal argument lists differ in length
1 error
```

Ex. 4.22

A compile error is thrown:

```
ConstructorExample3.java:15: error: constructor ObjA4 in class ObjA4 cannot be applied
to given types;
   {
   ^
  required: int
  found:    no arguments
  reason: actual and formal argument lists differ in length
1 error
```

Ex. 4.23

Here is the error:

```
ConstructorChaining_Ex_4_23.java:11: error: constructor A in class A cannot be applied
to given types;
class B extends A {
^
  required: int
  found:    no arguments
  reason: actual and formal argument lists differ in length
1 error
```

The default super() call in B tries to reach a no-paremeter constructor in A, which is not given. The compiler expects B to pass an int parameter to constructor A.

Ex. 4.24

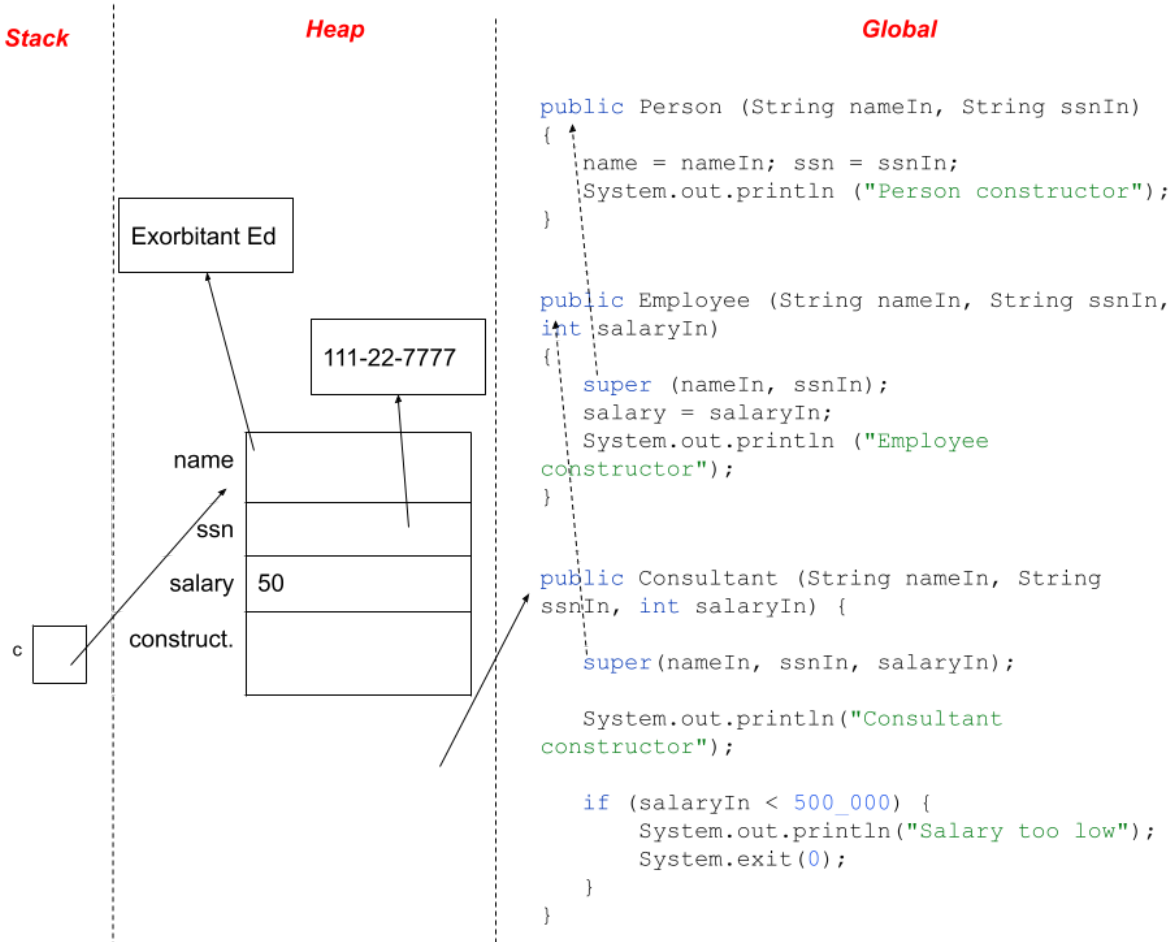
The output is

> Person constructor
> Employee constructor
> Consultant constructor
> Salary too low

Having received the call from main(), the constructor in Consultant calls the constructor in Employee with **super**. The constructor in Employee calls the constructor in Person with **super**. The constructor in Person assigns values to **name** and **ssn**, prints *"Person constructor"* and returns to the constructor in Employee, which assigns a value to **salary**, prints *"Employee constructor"* and returns to the constructor in Consultant, which prints *"Consultant constructor"* and having evaluating the salary as low prints *"Salary too low."* Now the execution returns to main().

Memory diagram on next page:

Ex. 4.24




```
public Person (String nameIn, String ssnIn)
{
    name = nameIn; ssn = ssnIn;
    System.out.println ("Person constructor");
}


public Employee (String nameIn, String ssnIn,
int salaryIn)
{
    super (nameIn, ssnIn);
    salary = salaryIn;
    System.out.println ("Employee
constructor");
}


public Consultant (String nameIn, String
ssnIn, int salaryIn) {

    super(nameIn, ssnIn, salaryIn);

    System.out.println("Consultant
constructor");

    if (salaryIn < 500_000) {
        System.out.println("Salary too low");
        System.exit(0);
    }
}
```

Ex. 4.25

The runtime exception is

> Exception in thread "main" java.lang.ClassCastException: class Person cannot be cast to class Customer425 (Person and Customer425 are in unnamed module of loader 'app')
> at TestPoly3.main(TestPoly3.java:21)

Ex. 4.27

Printed is

```
ObjA427@7ad041f3
ObjA427@7ad041f3
```

In other words, the keyword **this** used inside an object is a pointer to this very object (it allows the object to points to itself).

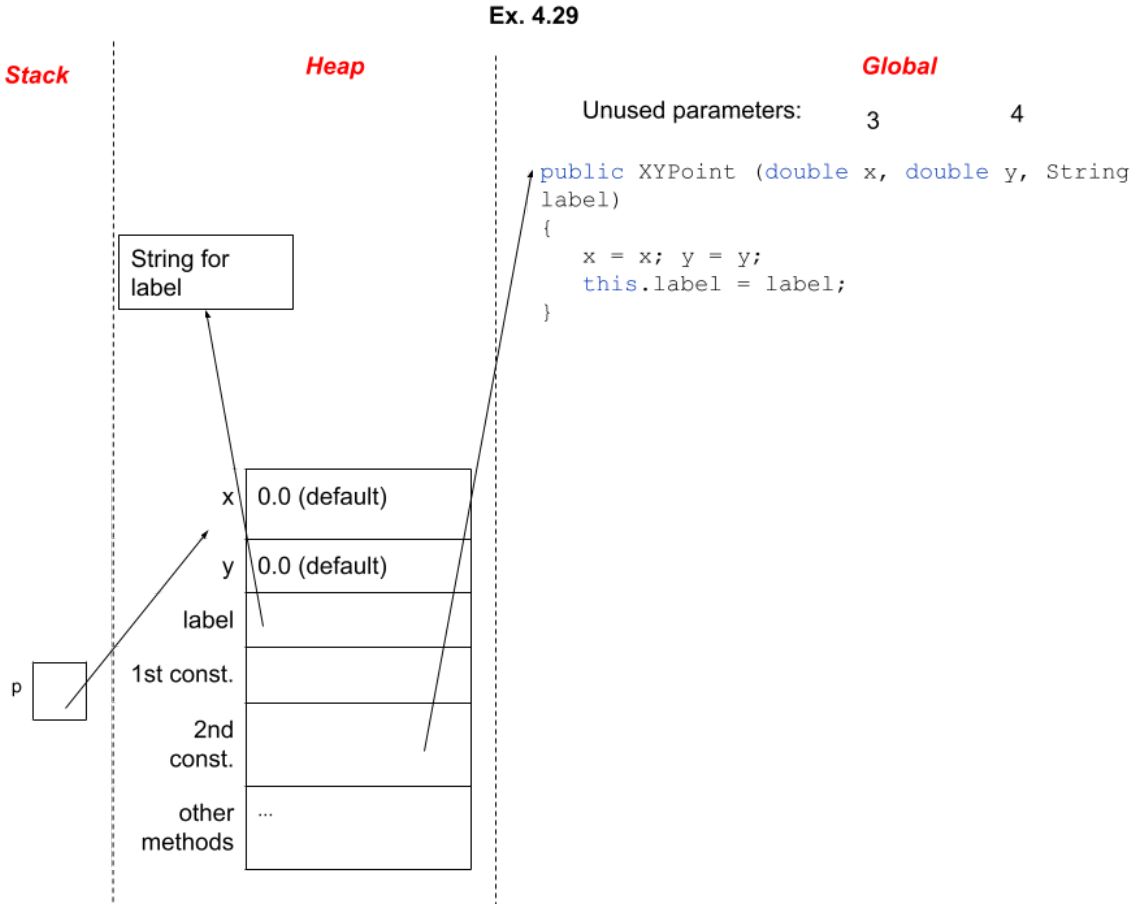
Ex. 4.29

With this code:

```
public XYPoint (double x, double y, String label)
{
  //this.x = x; this.y = y;
  x = x; y = y;
  this.label = label;
}
```

x and **y** are assigned a value of 0.0. This value is the default assigned to the instance variables x and y. Inside the constructor, x and y get assigned to themselves, while the values copied in the parameters are ignored (unused).

The memory diagram is below:



Ex. 4.31

The methods of class Object as listed at
https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html

- clone()
- equals(Object obj)
- finalize()
- getClass()
- hashCode()
- notify()
- notifyAll()
- toString()
- wait()
- wait(long timeout)

- wait(long timeout, int nanos)

Ex. 4.32

```
PreventInheritance.java:4: error: cannot inherit from final ObjV
class ObjW extends ObjV {
                   ^
1 error
```

Ex. 4.33

- In System.out.println(), **out** is **the variable name for a field** declared inside the **System** class: `public static final PrintStream out`. This variable is **static** (accessible only via the class name **System** and the dot operator), and it is **final** (it is constant; it cannot be changed). **out** is a variable of type **PrintStream,** so it allows access to methods defined inside PrintStream, via the dot operator, like so **out.println()**. So, **out** is a reference variable.
- The **println()** method can be found inside **the PrintStream class,** which can be found inside the j**ava.io package**. PrintStream defines ten (10) println() methods. Following are their signatures:
    - void println()
    - void println(boolean x)
    - void println(char x)
    - void println(char[] x)
    - void println(double x)
    - void println(float x)
    - void println(int x)
    - void println(long x)
    - void println(Object x)
    - void println(String x)