



CS-259

Data Structures with Java

Method Parameters

Instructor: **Joel Castellanos**

e-mail: joel@unm.edu

web: <http://cs.unm.edu/~joel/>

Course website:

<http://cs.unm.edu/~joel/cs259/>



9/18/2009

Quiz: Chapter 4 Reading

The Employee class shown below:

a) Is illegal because the public method `setSalary` changes a private field.

```
public class Employee
{ private String name;
  private double salary;
  public Employee(String n)
  { name = n;
  }
  public void setSalary(double s)
  { salary = s;
  }
  public double getSalary()
  { return salary;
  }
}
```

b) Is overly complex because the methods that read and write `salary` could be eliminated if `salary` was set to public.

c) Is illegal because the first method does not return a value.

d) Is illegal because `getSalary` does not accept any parameters.

e) Is basically ok.

2

Quiz: Which Line Has a Compile Error

```
1. public class Employee
2. { private String name;
3.   private double salary;
4.   public Employee(String n, double s)
5.   { name = n;
6.     salary = s;
7.   }
8.
9.   public void raiseSalary(double byPercent)
10.  { double raise = salary * byPercent/100.0;
11.    salary += raise;
12.  }
13.
14. public static void main(String[] args)
15. { Employee emp = new Employee("Tyal", 38000.0);
16.   raiseSalary(5.0);
17. }
18. }
```

3

a) line 4
b) line 9
c) line 10
d) line 11
e) line 16

Primitive Types: Passed By Value

```
1. public static int foo(int a, int b)
2. { a = a*a;
3.   b = b*b;
4.   return a+b;
5. }
6.
7.
8.
9. public static void main(String[] args)
10. {
11.   int a = 3;
12.   int b = 5;
13.   int c = foo(a,b);
14.   System.out.print(a + ", " + b + ", " + c);
15. }
```

Only the local copies of a and b change.

The values of a and b in main() are copied into new int memory locations.

Output: 3, 5, 34

4

Overloaded Method: sum()

```
public static int sum(int a, int b, int c)
{ System.out.print("sum(int, int, int): ");
  return a+b+c;
}
public static int sum(double a, double b, double c)
{ System.out.print("sum(double, double, double): ");
  //return a+b+c; //ERROR: Cannot convert double to int
  //return (int)a+b+c; //ERROR: Cannot convert double to int
  //return (int)a + (int)b + (int)c; //Ok, but different.
  return (int)(a+b+c);
}
public static void main(String[] args)
{ System.out.println( sum(4, 6, 7) );
  System.out.println( sum(4.4, 6.4, 7.4) );
}
```

```
sum(int, int, int): 17
sum(double, double, double): 18
```

5

What is Wrong with foo()



```
public int foo(boolean cool)
{ if (cool == true)
  { return 1;
  }
  if (cool == false)
  { return -1;
  }
}
```

```
public int foo(boolean cool)
{ if (cool == true)
  { return 1;
  }
  else
  { return -1;
  }
}
```

Compiler error Message:
"This method must return a
result of type int."

... but, it does ...
... doesn't it? ...

```
public int foo(boolean cool)
{ if (cool == true)
  { return 1;
  }
  return -1;
}
```

6

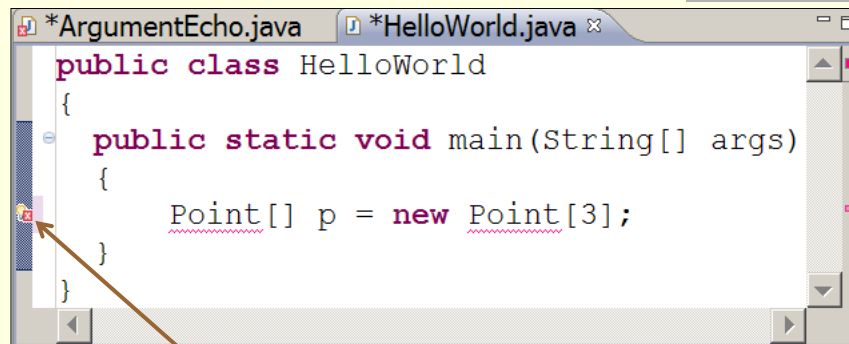
Quiz: Method return

```
public static int foo(boolean cool)
{ System.out.print("1 ");
  if (cool == true)
  { System.out.print("2 ");
    return 2;
  }
  else if (cool == false)
  { System.out.print("3 ");
    return 3;
  }
  System.out.print("4 ");
  return 4;
}
public static void main(String[] args)
{ System.out.println("["+foo(false)+"]");
}
```

- a) 1 3 [3]
- b) 1 2 3 [3]
- c) 1 3 3 [3]
- d) 1 3 4 [3]
- e) 1 3 4 [4]

7

Syntax Error: Eclipse Quick Fix



Right Click,
and Hold

- Toggle Breakpoint
- Disable Breakpoint
- Quick Fix** **Ctrl+1**
- Revert Line

8

Quick Fix: Part 2

From the *Quick Fix* dropdown menu, select *import Point*

```
import java.awt.Point;
public class HelloWorld
{
    public static void main(String[] args)
    {
        Point[] p = new Point[3];
    }
}
```

A **package** is the Java name for a library.

The **import** directive tells the compiler where to look for the class definitions when it comes upon a class that it cannot find in the default java.lang package.

9

JavaDoc: java.awt.Point

Fields x and y are public: no need to use get() / set().

Field Summary

int	x	The x coordinate.
int	y	The y coordinate.

Constructor Summary

Point ()	Constructs and initializes a point at the origin (0, 0) of the coordinate space.
Point (int x, int y)	Constructs and initializes a point at the specified (x, y) location in the coordinate space.
Point (Point p)	Constructs and initializes a point with the same location as the specified <code>Point</code> object.

Method: badSwap(int a, int b)

```
public class HelloWorld
{
    static void badSwap(int a, int b)
    {
        int temp = a;
        a = b;
        b = temp;
    }
    public static void main(String[] args)
    {
        int a = 5;
        int b = 7;
        System.out.println("a="+a + ", b="+b);
        badSwap(a, b);
        System.out.println("a="+a + ", b="+b);
    }
}
```

Output:

a=5, b=7

a=5, b=7

11

Method: swap(Point p)

```
static void swap(Point p)
{
    int temp = p.x;
    p.x = p.y;
    p.y = temp;
}

public static void main(String[] args)
{
    Point myPoint = new Point(5,7);
    System.out.println(
        "a="+myPoint.x + ", b="+myPoint.y);
    swap(myPoint);
    System.out.println(
        "a="+myPoint.x + ", b="+myPoint.y);
}
```

In Java, all non-primitive types (objects) are *passed by reference*.

Output:

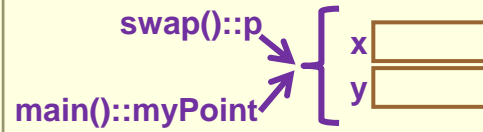
a=5, b=7

a=7, b=5

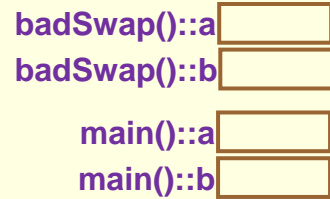
12

Method: swap(Point p)

```
static void swap(Point p)
{ int temp = p.x;
  p.x = p.y;
  p.y = temp;
}
```



```
static void badSwap(int a, int b)
{ int temp = a;
  a = b;
  b = temp;
}
```



```
public static void main(String[] args)
{ Point myPoint = new Point(5,7);
  int a = 5, b = 7;
  swap(myPoint);  badSwap(a, b);
}
```

13

Does objSwap() Swap?

```
public static void objSwap(Object a, Object b)
{ System.out.println("swap(): " + a + ", " + b);
  Object tmp=a;  a=b;  b=tmp;
  System.out.println("swap(): " + a + ", " + b);
}
```

```
public static void main(String[] args)
{ Integer a = new Integer(7);
  Integer b = new Integer(3);
  System.out.println("main(): " + a + ", " + b);
  objSwap(a, b);
  System.out.println("main(): " + a + ", " + b);
}
```

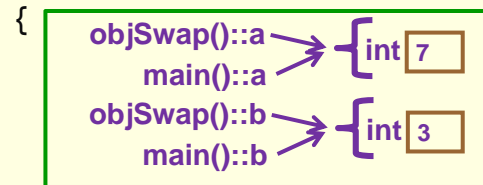
Output:

```
main(): 7, 3
swap(): 7, 3
swap(): 3, 7
main(): 7, 3
```

14

Analysis of objSwap()

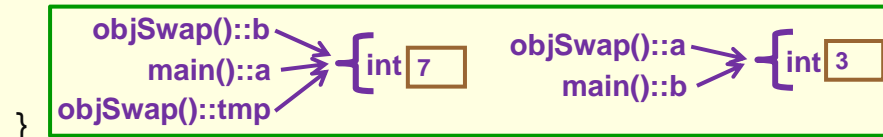
```
public static void objSwap(Object a, Object b)
```



This program creates:

- 2 Integer objects
- 5 Object references

```
Object tmp=a; a=b; b=tmp;
```



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

```
{ Integer a = new Integer(7);
  Integer b = new Integer(3);
  swap(a, b);
}
```

15

swap() VS objSwap()

```
public static void objSwap(Object a, Object b)
```

```
{ Object tmp = a;
  a = b;
  b = tmp; } ■ objSwap() swaps pointers.
```

```
public static void swap(Point p)
```

```
{ int temp = p.x;
  p.x = p.y;
  p.y = temp; } ■ swap() does not swap pointers.
```

■ swap() uses a pointer to reach inside an object and swap the object's data.

16

Allocating Space for Arrays of Objects

```
public static void main(String[] args)
{ //Allocates space for three 32-bit integers.
  int[] x = new int[3];

  //Allocates space for three references.
  Point[] p = new Point[3];

  for (int i = 0; i<3; i++)
  { x[i] = i; //No new

    //Create pointer object in the array.
    p[i] = new Point(i,i*2);
  }

  for (int i = 0; i<3; i++)
  { System.out.println(x[i] + ", " +
                      p[i].x + ", " + p[i].y);
  }
}
```

17

#1: What is the Output?

```
public class HelloWorld
{ public static void foo(Point tmp)
  { tmp.x = 5;
    tmp.y = 25;
  }

  public static void main(String[] args)
  { Point a = new Point(2,4);
    Point b = new Point(3,9);
    Point c = a;

    foo(a);

    System.out.println(a.x + ", " + a.y);
    System.out.println(b.x + ", " + b.y);
    System.out.println(c.x + ", " + c.y);
  }
}
```

18

#1: Answer

```
public class HelloWorld
{ public static void foo(Point tmp)
  { tmp.x = 5;
    tmp.y = 25;
  }

  public static void main(String[] args)
  { Point a = new Point(2,4);
    Point b = new Point(3,9);
    Point c = a;

    foo(a);

    System.out.println(a.x + ", " + a.y);
    System.out.println(b.x + ", " + b.y);
    System.out.println(c.x + ", " + c.y);
  }
}
```

2 a, c, tmp → Point(5,25)
b → Point(3,9)

1 a, c → Point(2,4)
b → Point(3,9)

3 5, 25
3, 9
5, 25

19

#2 What is the Output?

```
public class HelloWorld
{ public static void foo(Point tmp)
  { tmp = new Point(5,25);
  }

  public static void main(String[] args)
  { Point a = new Point(2,4);
    Point b = new Point(3,9);
    Point c = a;

    foo(a);

    System.out.println(a.x + ", " + a.y);
    System.out.println(b.x + ", " + b.y);
    System.out.println(c.x + ", " + c.y);
  }
}
```

20

#2 Answer

```
public class HelloWorld
{ public static void foo(Point tmp)
  { tmp = new Point(5,25);
  }
}
```

2 a, c → Point(2,4)
b → Point(3,9)
tmp → Point(5,25)

```
public static void main(String[] args)
{ Point a = new Point(2,4);
  Point b = new Point(3,9);
  Point c = a;
  foo(a);
}
```

1 a, c → Point(2,4)
b → Point(3,9)

```
System.out.println(a.x + ", " + a.y);
System.out.println(b.x + ", " + b.y);
System.out.println(c.x + ", " + c.y);
}
```

3 a, c → Point(2,4)
b → Point(3,9)
<garbage collect> Point(5,25)

21

#3 What is the Output?

```
public class HelloWorld
{ public static void foo(Point[] tmp)
  { tmp[1] = new Point(5,25);
  }

  public static void main(String[] args)
  { Point[] p = new Point[2];
    p[0] = new Point(2,4);
    p[1] = new Point(3,9);
    Point c = p[1];

    foo(p);

    System.out.println(p[0].x + ", " + p[0].y);
    System.out.println(p[1].x + ", " + p[1].y);
    System.out.println(c.x + ", " + c.y);
  }
}
```

22

#3 Answer

```
public class HelloWorld
{ public static void
    foo(Point[] tmp)
  { tmp[1] = new Point(5,25);
  }
}
```

2 p, tmp → { p[0], p[1] }
p[0] → Point(2,4)
c → Point(3,9)
p[1] → Point(5,25)

```
public static void main(String[] args)
{ Point[] p = new Point[2];
  p[0] = new Point(2,4);
  p[1] = new Point(3,9);
  Point c = p[1];
  foo(p);
}
```

1 p → { p[0], p[1] }
p[0] → Point(2,4)
p[1], c → Point(3,9)

3 p → { p[0], p[1] }
p[0] → Point(2,4)
c → Point(3,9)
p[1] → Point(5,25)

23