Software 1

Recitation No. 13
(Summary)

1

Initialization

```
public class Foo {
  static int bar;

  public static void main (String args []) {
     bar += 1;
     System.out.println("bar = " + bar);
  }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

2

Initialization

```
public class Test {
  private int a = getB();
  private int b = 5;

  private int getB() {
     return b;
  }
  public static void main(String args[]) {
     System.out.println((new Test()).a);
    }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

3

Initialization

```
public class Test {
  private int b = 5;
  private int a = getB();

  private int getB() {
     return b;
  }
  public static void main(String args[]) {
     System.out.println((new Test()).a);
    }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

4

Pass by Value

```
public class PassTest1{
  public static void changeInt(int value)
  {
     value = 55;
  }
  public static void main(String args[]) {
     int val = 11;
     changeInt(val);
     // What is the current value?
     System.out.println(val);
  }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

Pass by Value

```
public class PassTest2 {
  public static void changeObjectRef(MyPoint ref)
  {  ref = new MyPoint(1, 1);  }

  public static void main(String args[]){
    MyPoint point = new MyPoint(22,7);
    changeObjectRef(point);
    System.out.println(point);
  }
}
```

```
public class MyPoint {
   private int x, y;

   public MyPoint(int x, int y)
   { this.x = x;  this.y = y;}

   public String toString()
   { return "(" + x + ","
                + y + ")"; }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

6

Pass by Value

```
public class PassTest3 {

  public static void changeObjectAttr(MyPoint ref){
      ref.setX(4);
  }
  public static void main(String args[]) {
    MyPoint point = new MyPoint(22, 7);
    changeObjectAttr(point);
    // What is the current value?
    System.out.println(point);
  }
}
```

```
public class MyPoint {
   ...
   // new method
   public void setX(int x) {
        this.x = x;
   }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

Pass By-Value

```
public class Test {
  private static class Value { int v = 1; }

  public static void main(String[] args) {
     int v = 2;
    Value value = new Value();
    value.v = 3;
    foo(value, v);
    System.out.println(value.v + " " + v);
  }
  private static void foo(Value value, int v) {
    v= 4;
    value.v = 5;
    value =  new Value();
    System.out.println(value.v + " " + v);
  }
}
```

What is the output?

A Word about Interfaces

- An interface can extend several interfaces
- Interface methods are by definition public and abstract:

```
public interface MyInterface {
   public abstract int foo1(int i);
   int foo2(int i);
}
```

foo1 and foo2 have the same modifiers

Interfaces

```
public interface Foo {
  public void bar() throws Exception;
}

public class FooImpl implements Foo {
  public void bar() {
     System.out.println(“No exception is thrown");
  }

  public static void main(String args[]) {
     Foo foo = new FooImpl();
     foo.bar();
  }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

Interfaces

```
public interface Foo {
  public void bar()  throws Exception;
}

public class FooImpl implements Foo {
  public void bar() {
    System.out.println(“No exception is thrown");
  }

  public static void main(String args[]) {
     FooImpl foo = new FooImpl();
     foo.bar();
  }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

Interfaces and Inheritance

Consider the following class hierarchy:

```
Interface Animal {…}
class Dog implements Animal{…}
class Poodle extends Dog {…}
class Labrador  extends Dog {…}
```

Which of the following lines (if any) will not compile?

```
Poodle poodle = new Poodle();
Animal animal = (Animal)poodle;
Dog dog = new Labrador();
animal = dog;
poodle = dog;
```

Interfaces and Inheritance

```
class A {
    public void print() {
        System.out.println("A");
     }
}
interface C {
    void print();
}

class B extends A implements C {}
```

public by default

Does class B compile?

Interfaces and Inheritance

```
class A {
    void print() {
        System.out.println("A");
     }
}
interface C {
    void print();
}

class B extends A implements C {}
```

Does class B compile?

Inheritance

```
package a;
public class A {
   public void foo() {
      System.out.println("A.foo()");
   }
   public void bar() {
      System.out.println("A.bar()");
      foo();
   }
}
package b;
public class B extends A {
   public void foo() {
       System.out.println("B.foo()");
   }
   public static void main(String[] args) {
       A a = new B();
        a.bar();
   }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

Inheritance

```
package a;
public class A {
   void foo() {
      System.out.println("A.foo()");
   }
   public void bar() {
      System.out.println("A.bar()");
      foo();
   }
}
package b;
public class B extends A {
   public void foo() {
       System.out.println("B.foo()");
   }
   public static void main(String[] args) {
       A a = new B();
        a.bar();
   }
}
```

Does the code compile? If no, why?
Does the code throw a runtime exception?
If yes, why? If no, what is the output?

Inheritance

```
public class A {
        public void foo() {…}
}

public class B extends A {
  public void foo() {…}
}
```

How can you invoke the foo method of A within B?
Answer:
Use super.foo()

Inheritance

```
public class A {
  public void foo() {…}
}

public class B extends A {
  public void foo() {…}
}

public  class C extends B {
  public void foo() {…}
}
```

How can you invoke the foo method of A within C?
Answer:
Not possible
(super.super.foo() is illegal)

Inheritance & Constructors

```
public class A {
   String bar = "A.bar";
   A() { foo(); }
   public void foo() {
        System.out.println("A.foo(): bar = " + bar);
   }
}

public class B extends A {
   String bar = "B.bar";
   B() { foo(); }
   public void foo() {
        System.out.println("B.foo(): bar = " + bar);
   }
   public static void main(String[] args) {
           A a = new B();
           System.out.println(“a.bar = “ + a.bar);
           a.foo();
     }
}
```

What is the output?

Inheritance & Constructors

```
public class A {
   protected B b = new B();
   public A() { System.out.println("in A: no args."); }
   public A(String s) { System.out.println("in A: s = " + s); }
}

public class B {
   public B() { System.out.println("in B: no args."); }
}

public class C extends A {
   protected B b;
   public C() { System.out.println("in C: no args."); }
   public C(String s) { System.out.println("in C: s = " + s); }
}

public Class D {
   public static void main(String args[]) {
        C c = new C();
        A a = new C();
   }
}
```

What is the output?

Inheritance & Constructors

```
public class A {
   protected B b = new B();
   public A() { System.out.println("in A: no args."); }
   public A(String s) { System.out.println("in A: s = " + s); }
}

public class B {
   public B() { System.out.println("in B: no args."); }
}

public class C extends A {
   protected B b;
   public C() { System.out.println("in C: no args."); }
   public C(String s) { System.out.println("in C: s = " + s); }
}

public Class D {
   public static void main(String args[]) {
        C c = new C("c");
        A a = new C("a");
   }
}
```

What is the output?

Inheritance & Constructors

```
public class A {
   protected B b = new B();
   public A() { System.out.println("in A: no args."); }
   public A(String s) { System.out.println("in A: s = " + s); }
}

public class B {
   public B() { System.out.println("in B: no args."); }
}

public class C extends A {
   protected B b;
   public C() { System.out.println("in C: no args."); }
   public C(String s) { System.out.println("in C: s = " + s); }
}

public Class D {
   public static void main(String args[]) {
        C c = new C("c");
        A a = new C("a");
   }
}
```

What will happen if we remove this line?

Inheritance & Constructors

```
public class A {
   String bar = "A.bar";
}

public class B extends A {
   String bar = "B.bar";
   B() { foo(); }
   public void foo() {
        System.out.println("B.foo(): bar = " + bar);
   }
   public static void main(String[] args) {
        A a = new B();
        System.out.println(a.bar);
        a.foo();
   }
}
```

What is the result?

Overriding & Overloading

```
public class A{
  public int foo(Object o){return 0;}
}
public class B extends A{
  public int foo(Object o){return 1;}
  public int foo(String o){return 2;}
  public static void main(String[] args){
     A a = new B();
     System.out.println(a.foo(“hello”));
  }
}
```

what is the result?