Software 1

Recitation No. 13
(Summary)

Initialization

```
public class Foo {
    static int bar;

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

The output is:
bar = 1

mpile? If no, why?
ow a runtime exception?
If yes, why? If no, what is the output?

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);
    }
}
```

The output is:
0

compile? If no, why?
hrow a runtime exception?
If yes, why? If no, what is the output?

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);
    }
}
```

The output is:
5

compile? If no, why?
hrow a runtime exception?
If yes, why? If no, what is the output?

Pass by Value

```
public class PassTest1{

    public static void changeInt(int value) {
        value = 55;
    }

    public static void main(String args[]) {
        int val;

        // Assign the int
        val = 11;
        // Try to change it
        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?

The output is:
11

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;
     // Assign the point
     point = new MyPoint(22, 7);
     // Try to change it
     changeObjectRef(point);
     // What is the current value?
     System.out.println(point);
  }
}
```

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

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

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

The output is:
(22,7)

ompile? If no, why?
row a runtime exception?
, what is the output?

Pass by Value

```
public class PassTest3 {

  public static void changeObjectAttr(MyPoint ref){
            ref.setX(4);
  }
  public static void main(String args[]) {
     MyPoint point;
     // Assign the point
     point = new MyPoint(22, 7);

     changeObjectAttr(point);

     // What is the current value?
     System.out.println(point);
  }
}
```



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

      public MyPoint(int x, int y) {
            this.x = x;
            this.y = y;
      }
      public void setX(int x) {
            this.x = x;
      }
      @Override
      public String toString() {
            return "(" + x + "," + y + ")";
      }
}
```

The output is:
(4,7)

ompile? If no, why?
row a runtime exception?
, 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);
  }
}
```

The output is:
1 4
5 2

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);
}
```

The type of foo1 and foo2 is the same.

Interfaces

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

public class FooImpl implements Foo {
        public void bar() {
                System.out.println("An exception is not thrown");
        }

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

Compilation Error:
"Unhandled exception type Exception"

ption?
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();
    }
}
```

Output:
No exception is thrown

n?

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;
```

poodle = (Poodle) dog;
-No compilation error
-Runtime Exception

Labrador labrador = (Labrador) animal;
-No compilation error
-No Runtime Exception

Interfaces and Inheritance

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

class B extends A implements C {
}

interface C {
    void print();
}
```

No compilation errors

public by default

Interfaces and Inheritance

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

class B extends A implements C {
}

interface C {
    void print();
}
```

Compilation error:
The inherited package method A.print() cannot hide the public abstract method in C

Inheritance

```
public class A {
    public void foo() {
        System.out.println("A.foo()");
    }

    public void bar() {
        System.out.println("A.bar()");
        foo();
    }
}

public class B extends A {
    public void foo() {
        System.out.println("B.foo()");
    }
}
```

```
public class D {
    public static void main(String[] args) {
        A a = new B();
        a.bar();
    }
}
```

The output is:
A.bar()
B.foo()

ile? If no, why?
a runtime exception?
at is the output?

Inheritance

```
public class A {
    private void foo() {
        System.out.println("A.foo()");
    }

    public void bar() {
        System.out.println("A.bar()");
        foo();
    }
}

public class B extends A {
    public void foo() {
        System.out.println("B.foo()");
    }
}
```

```
public class D {
    public static void main(String[] args) {
        A a = new B();
        a.bar();
    }
}
```

The output is:
A.bar()
A.foo()

ile? If no, why?
a runtime exception?
at 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 class D {
    public static void main(String[] args) {
        A a = new B();
        System.out.println("a.bar = " + a.bar);
        a.foo();
    }
}
```

The output is:

B.foo(): bar = null
B.foo(): bar = B.bar
a.bar = A.bar
B.foo(): bar = B.bar

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();
        }
}
```

The output is:

in B: no args.
in A: no args.
in C: no args.
in B: no args.
in A: no args.
in C: no args.

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");
        }
}
```

The output is:

```
in B: no args.
in A: no args.
in C: s = c
in B: no args.
in A: no args.
in C: s = a
```

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");
        }
}
```

Compilation error without 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 class D {
        public static void main(String[] args) {
                A a = new B();
                System.out.println(a.bar);
                a.foo();
        }
}
```

What is the result?

undefined for the type A"