



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:	pile? If no, why?
<code>bar = 1</code>	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:	compile? If no, why?
0	throw 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? throw a runtime exception? 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 = 11;
        changeInt(val);
        // What is the current value?
        System.out.println(val);
    }
}
```

```
The output is:
11
```

compile? If no, why?
throw a runtime exception?
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);  
    }  
}
```

The output is: (22,7)
Does it compile? If no, why?
Does it throw a runtime exception?
If not, what is the output?

```
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 + ")";  
    }  
}
```

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);  
    }  
}
```

The output is: (4,7)
Does it compile? If no, why?
Does it throw a runtime exception?
If no, what is the output?

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

7

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);  
}
```

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();  
}
```

Compilation Error:
"Unhandled exception type Exception" option?
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");  
    }  
}
```

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

public by default

```
class B extends A implements C {}
```

No compilation errors

Interfaces and Inheritance

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

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

```
class B extends A implements C {}
```

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();
    }
}
```

The output is:

A.bar()

B.foo()

file? If no, why?

a runtime exception?

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();
    }
}
```

The output is:

A.bar()

A.foo()

file? If no, why?

a runtime exception?

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();
}
}
```

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); }  
}
```

Compilation error
without this line

```
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");  
    }  
}
```

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?

The method foo is
undefined for the type A"

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"));
    }
}
```

The result is: 1 | It?