

Software 1

Recitation No. 13 (Summary)

June 12, 2006

1

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
Does the code compile? If no, why? If yes, why? If no, what is the output? If yes, why? If no, what is the output?

June 12, 2006

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);
    }
}
```

The output is: 0
Does the code compile? If no, why? If yes, why? If no, what is the output? If yes, why? If no, what is the output?

June 12, 2006

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);
    }
}
```

The output is: 5
Does the code compile? If no, why? If yes, why? If no, what is the output? If yes, why? If no, what is the output?

June 12, 2006

4

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

June 12, 2006

5

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

June 12, 2006

6

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 + x + ", " + y + "n";
    }
}

```

The output is:
(4,7)

Compile? If no, why?
 Row a runtime exception?
 , what is the output?

June 12, 2006

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

8

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.

June 12, 2006

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" option?
 If yes, why? If no, what is the output?

10

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
 If yes, why? If no, what is the output?

11

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

June 12, 2006

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

June 12, 2006

13

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

June 12, 2006

14

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()

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

June 12, 2006

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()

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

June 12, 2006

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()

June 12, 2006

17

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)

June 12, 2006

18

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

June 12, 2006

19

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.

20

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

21

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

22

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"

June 12, 2006

23