

Software 1 with Java

Recitation No. 11 (Summary)

June 28, 2007

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

June 28, 2007

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

June 28, 2007

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

June 28, 2007

4

Initialization

```
public static void main(String args[]) {
    int i;
    System.out.println(i);
}
```

The code does not compile since the local variable i is not initialized.
If yes, why? If no, what is the output?

June 28, 2007

5

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);
        System.out.println(val);
    }
}
```

The output is: 11
If yes, why? If no, what is the output?
Compile? If no, why?
Throw a runtime exception?

June 28, 2007

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;
        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;
    }
}

```

The output is: 22,7

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

June 28, 2007 7

Pass by Value

```

public class PassTest3 {
    public static void changeObjectAttr(MyPoint ref) {
        ref.setX(4);
    }

    public static void main(String args[]) {
        MyPoint point;
        point = new MyPoint(22, 7);
        changeObjectAttr(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;
    }

    public void setX(int x) {
        this.x = x;
    }
}

```

The output is: 4,7

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

June 28, 2007

Pass by Value

```

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

    public static void main(String[] args) {
        Test test = new Test();
        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

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

June 28, 2007 9

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 28, 2007 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[]) {
        Foo foo = new FooImpl();
        foo.bar();
    }
}

```

Compilation Error: "Unhandled exception type Exception" option? If yes, why? If no, what is the output?

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

June 28, 2007

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

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

June 28, 2007

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 28, 2007

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 28, 2007

14

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 28, 2007

15

Method Overloading & Overriding

```
public class A {
    public float foo(float a, float b) throws IOException{
    }
}
```

```
public class B extends A {
    ...
}
```

Which of the following methods can be defined in B:

1. float foo(float a, float b){...}
2. public int foo(int a, int b) throws Exception{...}
3. public float foo(float a, float b) throws Exception{...}
4. public float foo(float p, float q) {...}

Answer: 2 and 4

16

Method Overriding

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

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

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

Casting is
unnecessary

The output is:
B
B
Compile? If no, why?
throw a runtime exception?
If yes, what is the output?

June 28, 2007

Method Overriding & Visibility

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

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

```
public class C {
    public static void main(String[] args) {
        B b = new B();
        b.print();
    }
}
```

Compilation error:
"Cannot reduce the visibility
of the inherited method"
no, why?
runtime exception?
the output?

June 28, 2007

18

Method Overriding & Visibility

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

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

```
public class C {
    public static void main(String[] args) {
        B b = new B();
        b.print();
    }
}
```

The output is:

June 28, 2007

19

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

The output is:

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

June 28, 2007

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

The output is:

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

June 28, 2007

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 28, 2007

22

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 28, 2007

23

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 28, 2007

24

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.

25

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

26

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?

27

Good-Luck!!!

June 28, 2007

28