

Software 1 with Java

Recitations No. 13-14 (Summary)

August 20, 2006

Oranit Dror

1

Singleton

- Ensures a class has only one instance and provides a global access point to it.

```
public class Logger {  
    private static final Logger instance = new Logger();  
  
    private Logger() {...}  
  
    public static Logger getInstance() {  
        return instance;  
    }  
}
```

August 20, 2006

Oranit Dror

2

Singleton (cont.)

```
public class Logger {  
    private static Logger instance;  
  
    private Logger() {...}  
  
    public static Logger getInstance() {  
        if (instance == null)  
            instance = new Logger();  
  
        return instance;  
    }  
}
```

Lazy evaluation

August 20, 2006

Oranit Dror

3

Enumeration

- Enforces a final set of instances and provides a global access point to them.

```
public final class Boolean ... {  
    public static final Boolean FALSE = new Boolean(false);  
    public static final Boolean TRUE = new Boolean(true);  
  
    // Constructor  
    public Boolean(boolean value) {...}  
    // Factory Method  
    public static Boolean valueOf(boolean b) {...}  
    ...  
}
```

August 20, 2006

Oranit Dror

4

Enumeration (cont.)

```
public final class Boolean ... {  
    public static final Boolean FALSE = new Boolean(false);  
    public static final Boolean TRUE = new Boolean(true);  
  
    public Boolean(boolean value) {...}  
  
    static Boolean valueOf(boolean b) {  
        return (b ? Boolean.TRUE : Boolean.FALSE);  
    }  
}
```

private
is better

August 20, 2006

Oranit Dror

5

Abstract Factory

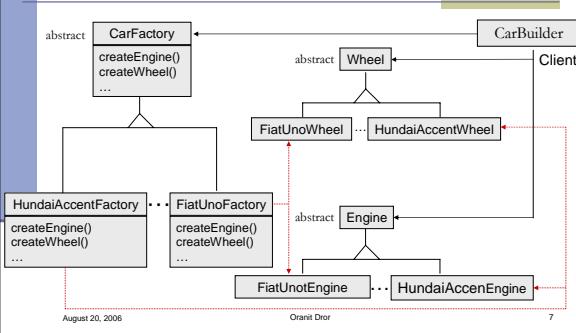
- Useful for creating families of related objects without specifying their concrete classes
- Example: An application for building cars
 - builds various types of cars:
Hundai-Accent, Peugeot 205 GTI, Fiat-Uno etc.
 - all cars have the same overall structure,
i.e. consist of the same components:
engine, wheels, brakes etc.
 - The components are different.

August 20, 2006

Oranit Dror

6

Abstract Factory (cont.)



Abstract Factory (cont.)

- Isolates concrete classes
- Exchanging product families is easy
- Promotes consistency among products
- Supporting new kinds of products involves changing the AbstractFactory class and all of its subclasses.
- Typically implemented as a singleton.

August 20, 2006

Oranit Dror

8

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?

August 20, 2006

Oranit Dror

9

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?

August 20, 2006

Oranit Dror

10

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?

August 20, 2006

Oranit Dror

11

Initialization

```

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

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

August 20, 2006

Oranit Dror

12

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);  
    }  
}
```

August 20, 2006

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

Oranit Dror

13

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);  
    }  
}
```

August 20, 2006

Oranit Dror

14

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 + ")";
 }
}

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 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);  
    }  
}
```

August 20, 2006

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

Oranit Dror

15

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);  
    }  
}
```

What is the output?

16

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.

August 20, 2006

Oranit Dror

17

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();  
    }  
}
```

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

18

Interfaces

```
public interface Foo {  
    public void bar()  
        throws Exception;  
}  
  
public class FooImpl implements Foo {  
    public void bar0() {  
        System.out.println("No exception is thrown");  
    }  
  
    public static void main(String args[]) {  
        FooImpl foo = new FooImpl();  
        foo.bar0();  
    }  
}
```

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

19

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;
```

August 20, 2006

Oranit Dror

20

Interfaces and Inheritance

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

Is there any error?

August 20, 2006

Oranit Dror

21

Interfaces and Inheritance

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

Is there any error?

August 20, 2006

Oranit Dror

22

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();  
    }  
}
```

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

August 20, 2006

Oranit Dror

23

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();  
    }  
}
```

What is the output?

August 20, 2006

Oranit Dror

24

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

25

Method Overriding

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

August 20, 2006

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

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() {  
        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()");  
    }  
}
```

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

August 20, 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()");  
    }  
}
```

August 20, 2006

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?

August 20, 2006

Ornit Dror

29

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?

August 20, 2006

Ornit Dror

30

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);
    }
}
```

August 20, 2006

Oranit Dror

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

What is the output?

31

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?

32

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");
    }
}
```

33

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?

34

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);
    }
}
```

August 20, 2006

Oranit Dror

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

What is the result?

35

Local Class

```
public class Test {
    public int a = 0;
    private int b = 1;

    public void foo(final int c) {
        int d = 2;

        class InnerTest {
            private void bar(int e) {
                System.out.println("e = " + e);
            }
        }
    }
}
```

Which variables (a, b, c, d, e) are accessible at the highlighted line?

36

