Advanced Programming with Java

Summary Solutions

Ohad Barzilay

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?

Exceptions

```
int i=1, j=1;
try {
    i++;
     j--;
     if (i/j > 1)
          i++;
} catch(ArithmeticException e) {
    System.out.println(1);
} catch(Exception e) {
    System.out.println(2);
} finally {
    System.out.println(3);
}
```

The output is:
1
3

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 {
      int x;
      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 {
     int x;
     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?

Homework

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?

String Interning

```
public static void main(String args[]) {
    if ("hello".substring(0) == "hello")
            System.out.println("statement 1 is true");

    if ("hello".substring(1) == "ello")
            System.out.println("statement 2 is true");

    if ("hello".replace('l','l') == "hello")
            System.out.println("statement 3 is true");

    if ("hello".replace('h','H') == "hello".replace('h','H'))
            System.out.println("statement 4 is true");

    if ("hello".replace('h','H') == "Hello")
            System.out.println("statement 5 is true");
}
```

The output is:

statement 1 is true

statement 3 is true

String Interning

```
public static void main(String args[]) {
    String s1 = "he";
    String s2 = "he" + "llo";
    String s3 = s1 + "llo";

    if (s2.equals(s3))
        System.out.println("statement 1 is true");
    if (s2 == "hello")
        System.out.println("statement 2 is true");
    if (s2 == s3)
        System.out.println("statement 3 is true");
    if (s2 == new String("hello"))
        System.out.println("statement 4 is true");
    if (s2 == s3.intern())
        System.out.println("statement 5 is true");
    if (s3 == s2.intern())
        System.out.println("statement 6 is true");
}
```

The output is:

statement 1 is true

statement 2 is true

statement 5 is true

Visibility

```
public class A {
    private int bar =0;

    public boolean isEqual(A a) {
        return (bar == a.bar);
    }

    public static void main(String[] args) {
        A a1 = new A();
        A a2 = new A();
        System.out.println(a1.isEqual(a2));
    }
}
```

The output is:
true

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

No compilation error:

Objects of the same class can access each other's private fields

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();

poodle = dog;

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

poodle = (Poodle) 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();
}
```

Is there any error?

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();
}
```

Is there any error?

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

Method Overloading

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

    public void foo(String s) {
        System.out.println("String");
    }

    public static void main(String args[]) {
        A a = new A();
        a.foo(null);
    }
}
```

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

Answer: The code compiles and runs, printing “String”

Method Overloading

```
public class A {
    private static class B {}
    private static class C extends B {}
    public void foo(B b) {
        System.out.println("B");
    }
    public void foo(C c) {
        System.out.println("C");
    }
    public static void main(String args[]) {
        A a = new A();
        a.foo(null);
    }
}
```

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

Answer: The code compiles and runs, printing “C”

Method Overloading

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

    public void foo(String s) {
        System.out.println("String");
    }

    public static void main(String args[]) {
        A a = new A();
        a.foo(null);
    }
}
```

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

Answer: The code does not compile (an ambiguous method)

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 unneeded

The output is:
B
B

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

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"

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

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?

The output is:
B

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()

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?

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."); }
}

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 {
        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?

Compilation Error:
"The method foo is undefined for the type A"

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) {
                [highlighted line]
            }
        }
    }
}
```

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

Only a,b,c and e are accessible at the highlighted line.

Good-Luck!!!