Programming Language II
CSE-215

Prof. Dr. Mohammad Abu Yousuf

yousuf@juniv.edu

Find Output

```
What will be the output?

// filename Test.java
class Test {
    public static void main(String[] args) {
        for(int i = 0; 1; i++) {
            System.out.println("Hello");
            break;
        }
    }
}
```

```
What will be the output?
// filename Test.java
class Test {
    public static void main(String[] args) {
        for(int i = 0; 1; i++) {
            System.out.println("Hello");
            break;
        }
    }
}
```

Output: Compiler Error

There is an error in condition check expression of for loop. Java differs from C++(or C) here. C++ considers all non-zero values as true and 0 as false. Unlike C++, an integer value expression cannot be placed where a boolean is expected in Java. Following is the corrected program.

```
// filename Test.java
class Test {
    public static void main(String[] args)
{
        for(int i = 0; true; i++) {
            System.out.println("Hello");
            break;
        }
    }
}
// Output: Hello
```

```
What will be the output?

// filename Main.java
class Main {
    public static void main(String args[])
{
        System.out.println(fun());
    }
    int fun() {
        return 20;
    }
}
```

```
What will be the output?

// filename Main.java
class Main {
    public static void main(String args[])
{
        System.out.println(fun());
    }
    int fun() {
        return 20;
    }
}
```

Output: Compiler Error

Like C++, in Java, non-static methods cannot be called in a static method. If we make fun() static, then the program compiles fine without any compiler error. Following is the corrected program.

```
// filename Main.java
class Main {
   public static void main(String args[])
{
        System.out.println(fun());
    }
    static int fun() {
        return 20;
    }
}
// Output: 20
```

What will be the output?

```
// filename Test.java
class Test {
   public static void main(String args[])
{
       System.out.println(fun());
   }
   static int fun() {
       static int x= 0;
       return ++x;
   }
}
```

What will be the output?

```
// filename Test.java
class Test {
   public static void main(String args[])
{
       System.out.println(fun());
   }
   static int fun() {
       static int x= 0;
       return ++x;
   }
}
```

Output: Compiler Error

Unlike C++, static local variables are not allowed in Java. We can have class static members (instance variable). Following is the corrected program:

```
class Test {
   private static int x;
  public static void main(String args[])
{
       System.out.println(fun());
   }
   static int fun() {
       return ++x;
   }
}
// Output: 1
```

What will be the output of the program?

```
class Super
{
    public int i = 0;
    public Super(String text) /* Line 4 */
    {
        i = 1;
    }
}
class Sub extends Super
{
    public Sub(String text)
    {
        i = 2;
    }
    public static void main(String args[])
    {
        Sub sub = new Sub("Hello");
        System.out.println(sub.i);
    } }
```

What will be the output of the program?

```
class Super
{
    public int i = 0;
    public Super(String text) /* Line 4 */
    {
        i = 1;
    }
}
class Sub extends Super
{
    public Sub(String text)
    {
        i = 2;
    }
    public static void main(String args[])
    {
        Sub sub = new Sub("Hello");
        System.out.println(sub.i);
    }}
```

Compilation fails.

Explanation:
A default no-args constructor is not created because there is a constructor supplied that has an argument, line 4. Therefore the sub-class constructor must explicitly make a call to the super class constructor:

```
public Sub(String text) { super(text);
// this must be the first line
constructor i = 2; }
```

What will be the output of the program?

```
interface Count
{
    short counter = 0;
    void countUp();
}
public class TestCount implements Count
{
    public static void main(String [] args)
    {
        TestCount t = new TestCount();
        t.countUp();
    }
    public void countUp()
    {
        for (int x = 6; x>counter; x--, ++counter) /* Line 14 */
        {
            System.out.print(" " + counter);
        }
    }}
```

What will be the output of the program?

```
interface Count
{
   short counter = 0;
   void countUp();
}
public class TestCount implements Count
{
   public static void main(String [] args)
   {
      TestCount t = new TestCount();
      t.countUp();
   }
   public void countUp()
   {
      for (int x = 6; x>counter; x--, ++counter) /* Line 14 */
      {
         System.out.print(" " + counter);
      }
   }}
```

Answer: Compilation fails

Explanation:
The code will not compile because the variable counter is an interface variable that is by default `final static`. The compiler will complain at line 14 when the code attempts to increment counter.

What will be the output of the program?

```
import java.util.*;
public class NewTreeSet2 extends NewTreeSet
{
    public static void main(String [] args)
    {
        NewTreeSet2 t = new NewTreeSet2();
        t.count();
    }
}
protected class NewTreeSet   // Line 10
{
    void count()
    {
        for (int x = 0; x < 7; x++,x++ )
        {
            System.out.print(" " + x);
        }
    }
}
```

What will be the output of the program?

```
import java.util.*;
public class NewTreeSet2 extends NewTreeSet
{
    public static void main(String [] args)
    {
        NewTreeSet2 t = new NewTreeSet2();
        t.count();
    }
}
protected class NewTreeSet    // Line 10
{
    void count()
    {
        for (int x = 0; x < 7; x++,x++ )
        {
            System.out.print(" " + x);
        }
    }
}
```

Answer: Compilation fails at line 10

Explanation:
Nonnested classes cannot be marked `protected` (or `final` for that matter), so the compiler will fail at `protected class NewTreeSet`

What will be the output of the program?

```
package foo;
import java.util.Vector; /* Line 2 */
private class MyVector extends Vector
{
   int i = 1; /* Line 5 */
   public MyVector()
   {
      i = 2;   }
}
public class MyNewVector extends MyVector
{
   public MyNewVector ()
   {
      i = 4; /* Line 15 */
   }
   public static void main (String args [])
   {
      MyVector v = new MyNewVector(); /* Line 19 */
   } }
```

What will be the output of the program?

```
package foo;
import java.util.Vector; /* Line 2 */
private class MyVector extends Vector
{
    int i = 1; /* Line 5 */
    public MyVector()
    {
        i = 2;   }
}
public class MyNewVector extends MyVector
{
    public MyNewVector ()
    {
        i = 4; /* Line 15 */
    }
    public static void main (String args [])
    {
        MyVector v = new MyNewVector(); /* Line 19 */
    } }
```

Answer: Compilation will fail at line 3.

Explanation:

The compiler complains with the error "modifier private not allowed here". The class is created private and is being used by another class on line 19.

What will be the output of the program?

```java
// file name: Main.java

class Base {
    protected void foo() {}
}
class Derived extends Base {
    void foo() {}
}
public class Main {
    public static void main(String args[]) {
        Derived d = new Derived();
        d.foo();
    }
}
```

What will be the output of the program?

```java
// file name: Main.java

class Base {
    protected void foo() {}
}
class Derived extends Base {
    void foo() {}
}
public class Main {
    public static void main(String args[]) {
        Derived d = new Derived();
        d.foo();
    }
}
```

Output: Compiler Error

foo() is protected in Base and default in Derived. Default access is more restrictive. When a derived class overrides a base class function, more restrictive access can't be given to the overridden function. If we make foo() public, then the program works fine without any error. The behavior in C++ is different. C++ allows to give more restrictive access to derived class methods.

What will be the output of the program?

```java
final class A
{
   int i;
}
class B extends A
{
  int j;
  System.out.println(j + " " + i);
}
class inheritance
{
 public static void main(String args[])
 {
  B obj = new B();
  obj.display();
 }
}
```

What will be the output of the program?

```
final class A
{
   int i;
}
class B extends A
{
  int j;
  System.out.println(j + " " + i);
}
class inheritance
{
 public static void main(String args[])
 {
  B obj = new B();
  obj.display();
 }
}
```

Answer: **Compilation Error**

Explanation: class A has been declared final hence it cannot be inherited by any other class. Hence class B does not have member i, giving compilation error.

What will be the output of the program?

```
interface calculate
{
   void cal(int item);
}
class display implements calculate
{
  int x;
  public void cal(int item)
  {
     x = item * item;
  }
}
class interfaces
{
 public static void main(String args[])
 {
   display arr = new display;
   arr.x = 0;
   arr.cal(2);
   System.out.print(arr.x);
 }
}
```

What will be the output of the program?

```java
interface calculate
{
   void cal(int item);
}
class display implements calculate
{
  int x;
  public void cal(int item)
  {
     x = item * item;
  }
}
class interfaces
{
 public static void main(String args[])
 {
   display arr = new display;
   arr.x = 0;
   arr.cal(2);
   System.out.print(arr.x);
 }
}
```

Answer: **4**

What will be the output of the program?

```
class A
{
  public int i;
  protected int j;
}
class B extends A
{
   int j;
   void display()
  {
     super.j = 3;
     System.out.println(i + " " + j);
  }
}
class Output
{
  public static void main(String args[])
  {
     B obj = new B();
     obj.i=1;
     obj.j=2;
     obj.display();
}}
```

Answer: 1 2

Explanation: Both class A & B have member with same name that is j, member of class B will be called by default if no specifier is used. i contains 1 & j contains 2. So output is 1 2.

What will be the output of the program?

```java
class A
{
  public int i;
  protected int j;
}
class B extends A
{
   int j;
   void display()
  {
     super.j = 3;
     System.out.println(i + " " + j);
  }
}
class Output
{
  public static void main(String args[])
  {
     B obj = new B();
     obj.i=1;
     obj.j=2;
     obj.display();
}}
```

What will be the output of the program?

```java
class A
{
   public int i;
   private int j;
}
class B extends A
{
   void display()
   {
     super.j = super.i + 1;
     System.out.println(super.i + " " + super.j);
  }
}
class inheritance
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

What will be the output of the program?

```
class A
{
   public int i;
   private int j;
}
class B extends A
{
   void display()
   {
     super.j = super.i + 1;
     System.out.println(super.i + " " + super.j);
  }
}
class inheritance
{
    public static void main(String args[])
    {
        B obj = new B();
        obj.i=1;
        obj.j=2;
        obj.display();
    }
}
```

Answer: **Compilation Error**

Explanation: Class contains a private member variable j, this cannot be inherited by subclass B and does not have access to it.

Thank you