SWEN221: Software Development

13: Java Puzzlers

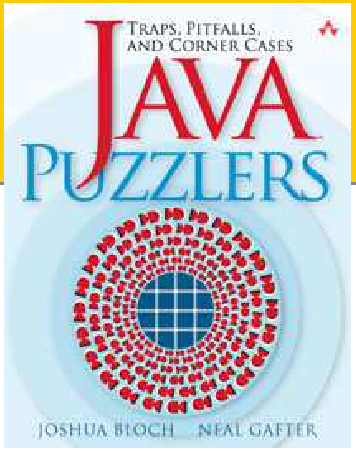
David J. Pearce & Nicholas Cameron & James Noble
Engineering and Computer Science, Victoria University

Java Puzzlers

How well do you know Java?

(See "Java Puzzlers", Addison Wesley)

About Java



- Java
  - It's a **complicated** language!
  - Most programmers (even really good ones) **don't know all the rules**

- Java Language Specification (JLS)
  - Provides a (nearly) complete **guide to the rules**.
    - See:

http://java.sun.com/docs/books/jls/third_edition/html/j3TOC.html

Puzzle #1 (Division)

- What does this code print?

```
int x = (-1 / 2);
int y = (1 / 2);

System.out.println(x + “,” + y);
```

A) 0,1   B) -1,0   C) 0, 0

Puzzle #1 (Division)

- What does this code print?

```
int x = (-1 / 2);
int y = (1 / 2);

System.out.println(x + “,” + y);
```

A) 0,1 ✗ B) -1,0 ✗ C) 0, 0 ✓

Because: Java always rounds towards zero (for ints), see JLS 15.17.2

Puzzle #2 (Post Increment)

- What does this code print?

```
int x = 0;
int y = x++ + x++ + x++;

System.out.println(y);
```

A) 0 B) 2 C) 3

Puzzle #2 (Post Increment)

- What does this code print?

```
int x = 0;
int y = x++ + x++ + x++;

System.out.println(y);
```

A) 0 ✗ B) 2 ✗ C) 3 ✓

Puzzle #3 (oddity)

- How to check an integer is odd?

```
boolean isOdd(int x) {
 return (x%2) == 1;
}
```

  - Does this method work?

A) Yes B) No C) Don't know

Puzzle #3 (oddity)

- How to check an integer is odd?

```
boolean isOdd(int x) {
 return (x%2) == 1;
}
```

  - Does this method work?

A) Yes ✗ B) No ✓ C) Don't know ✗

Because: `(-1 % 2) == -1` (in Java)

Puzzle #4 (Binary Operators)

- What does this code print?

```
int x = 3 * 11 / 2;
int y = 11 / 2 * 3;
System.out.println(x + “,” + y);
```

A) 15,16 B) 16,1 C) 16,15

Puzzle #4 (Binary Operators)

- What does this code print?

```
int x = 3 * 11 / 2;
int y = 11 / 2 * 3;
System.out.println(x + “,” + y);
```

A) 15,16 ✗　　B) 16,1 ✗　　C) 16,15 ✓

Because: * and / have same precedence, so Java executes them in left to right order!

Puzzle #5 (Finally)

- What does this code print?

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

static boolean f() {
 try { return true; }
 finally { return false; }
}
```

A) true B) false C) doesn't compile

Puzzle #5 (Finally)

- What does this code print?

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

static boolean f() {
 try { return true; }
 finally { return false; }
}
```

A) true ✗ [✓ B) false] C) doesn't compile ✗

Because: finally always comes last!

Puzzle #6 (Exceptions)

- What does this code print?

```
try {
 try {
   String x = null;
   x.toString();
 } catch(NullPointerException e1) {
   int x = 10 / 0;
 } catch(ArithmeticException e2) {
   System.out.println(“1”);
 }
} catch(ArithmeticException e2) {
 System.out.println(“2”);
}
```

A) 1 B) 2 C) other

Puzzle #6 (Exceptions)

- What does this code print?

```
try {
 try {
   String x = null;
   x.toString();
 } catch(NullPointerException e1) {
   int x = 10 / 0;
 } catch(ArithmeticException e2) {
   System.out.println("1");
 }
} catch(ArithmeticException e2) {
 System.out.println("2");
}
```

A) 1 ✗   B) 2 ✓   C) other ✗

Puzzle #7 (Constructors)

- What does this code print?

```
public class Test {
    Test() { f(); }
    void f() {}
}

public class Test2 extends Test {
    int i = 1;
    void f() { System.out.println(i); }

    public static void main(String[] args) {
       new Test2();
}   }
```

A) 0 B) 1 C) nothing

Puzzle #7 (Constructors)

- What does this code print?

```
public class Test {
    Test() { f(); }
    void f() {}
}

public class Test2 extends Test {
    int i = 1;
    void f() { System.out.println(i); }

    public static void main(String[] args) {
       new Test2();
}   }
```

[✓ A) 0]   B) 1 ✗   C) nothing ✗

Because: super constructor called before field initialisation!

Puzzle #8 (Multiplication)

- What does this code print?

```
public class Test {
    public static void main(String[] args) {
      int x = 60 * 60 * 24 * 1000 * 1000;

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

A) 86400000000000 B) 1 C) other

Puzzle #8 (Multiplication)

- What does this code print?

```
public class Test {
    public static void main(String[] args) {
      int x = 60 * 60 * 24 * 1000 * 1000;

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

A) 864000000000 ✗ B) 1 ✗ [✓ C) other]

Because: `integer overflow!`

Actually prints: `500654080`

Puzzle #9 (Sums)

- What does this code print?

```
int[] arr = {77, 077, 0x4D};
int sum = 0;

for(int i : arr) {
 sum = sum + i;
}

System.out.println(sum);
```

A) 232 B) 231 C) 217

Puzzle #9 (Sums)

- What does this code print?

```
int[] arr = {77, 077, 0x4D};
int sum = 0;

for(int i : arr) {
 sum = sum + i;
}

System.out.println(sum);
```

77 = 77
077 = 63
0x4D = 77
= 217

A) 232 ✗ B) 231 ✗ C) 217 ✓

Puzzle #10 (Static Blocks)

- What does this code print?

```
public class Test {
    static Test t1 = new Test();
    static Integer t2 = new Integer(1);

    Integer i1;

    public Test() { i1 = t2; }
    int f() { return i1; }

    public static void main(String[] args) {
      System.out.println(t1.f());
}   }
```

A) 1 B) 0 C) other

Puzzle #11 (Static Blocks)

- What does this code print?

```
public class Test {
    static Test t1 = new Test();
    static Integer t2 = new Integer(1);

    Integer i1;

    public Test() { i1 = t2; }
    int f() { return i1; }

    public static void main(String[] args) {
      System.out.println(t1.f());
}   }
```

A) 1 ✗ B) 0 ✗ C) other ✓

Puzzle #11 (Final)

```
public class Final {
 public Final() { trickster(); }
 void trickster() {}

 public static class Inner extends Final {
  public int x,y = 123;
  public final int z = 456;

  public void Inner() { x += 10; }
  void trickster() { x += y + z; }
 }

 public static void main(String[] args) {
  System.out.println(new Inner().x);
}}
```

A) 589 B) 466 C) 456 d) 123 e) 579

Puzzle #11 (Final)

```
public class Final {
 public Final() { trickster(); }
 void trickster() {}

 public static class Inner extends Final {
  public int x,y = 123;
  public final int z = 456;

  public void Inner() { x += 10; }
  void trickster() { x += y + z; }
 }

 public static void main(String[] args) {
  System.out.println(new Inner().x);
}}
```

A) 589 B) 466 [✓ C) 456] d) 123 e) 579

Puzzle #12 (Equality)

What does this code print?

```
public class FarmYard {
public static void main(String[] a) {
final String pig = "length: 10";
final String dog = "length: " + pig.length();
System.out.println("Animals are equal: " +
pig == dog);
}
}
```

A) "length: true" B) "length: false" C) other

Puzzle #12 (Equality)

What does this code print?

```
         public class FarmYard {
   public static void main(String[] a) {
       final String pig = "length: 10";
final String dog = "length: " + pig.length();
 System.out.println("Animals are equal: " +
                          pig == dog);
                     }
                    }
```

A) "length: true" ✗ B) "length: false" ✗ [✓ C) other]