Effective Java Puzzlers

Christoph Pickl, JSUG, 12th January 2009

Your Name: _____

Correct Answers: _____

1 Simple Subtraction [2]

Imagine you want to write a little checkbook app for your own. It should only do the fundamental basics without any unnecessary overhead. Here is a programm that tries to solve a really really simple problem. What does it print?

```
public class SimpleSubtraction {
  public static void main(String[] args) {
    System.out.println(2.00 - 1.10);
  }
}
```

☐ 0.90 ☐ 0.9 ☐ .9

☐ 0.8999999999999999 ☐ Does not compile

☐ Other: _____

2 Simple Addition [4]

Of course because we are all working for a financial institute and we don't want only to subtract things (like from our salary), we also want to add some amount to another. So let's do it: What does this single statement print?

```
public class SimpleAddition {
  public static void main(String[] args) {
    System.out.println(12345 + 5432l);
  }
}
```

☐ 55555 ☐ 66666 ☐ 17777 ☐ Does not compile

☐ Other: _____

3 Simple Division [3]

The first two puzzles dealed with elementary arithemtic. Now let's go to some more complex stuff: Division. The dividend represents the number of microseconds in a day; the divisor, the number of milliseconds in a day. What does the program print?

```
public class SimpleDivision {
  public static void main(String[] args) {
    final long MICROS_PER_DAY = 24 * 60 * 60 * 1000 * 1000;
    final long MILLIS_PER_DAY = 24 * 60 * 60 * 1000;
    System.out.println(MICROS_PER_DAY / MILLIS_PER_DAY);
  }
}
```

☐ 1 ☐ 5 ☐ 1000 ☐ Does not compile

☐ Other: _____

4 Compound Legal [9]

Now it's your turn to write some code. Simply provide declarations for the variables `x` and `i` such that the first statement is legal and the second is illegal.

```
{ // first statement legal
  _________ x = _________;
  _________ i = _________;
  x += i;
}
{ // second statement illegal
  _________ x = _________;
  _________ i = _________;
  x = x + i;
}
```

5 Compound Illegal [10]

This puzzle is very similar to the last one, except that the first statement should be made illegal and the second legal.

```
{ // first statement illegal
  _________ x = _________;
  _________ i = _________;
  x += i;
}
{ // second statement legal
  _________ x = _________;
  _________ i = _________;
  x = x + i;
}
```

6 Unicode Escapes

The following program uses two Unicode escapes, which represent Unicode characters by their hexadecimal numeric codes. What does the program print?

```
public class EsacpeRout {
  public static void main(String[] args) {
    // \u0022 is the unicode escape for double quote (")
    System.out.println("a\u0022.length()+\u0022b".length());
  }
}
```

☐ 2 ☐ 14 ☐ 24 ☐ Does not compile

☐ Other: _____

7 Classify Characters [19]

Let's continue with some string operations. This piece of software detects of what kind the given character is. What does it print?

```
public class Classifier {
  public static void main(String[] args) {
    System.out.println(classify('n') +
                       classify('+') + classify('2'));
  }
  public static String classify(char c) {
    if("0123456789".indexOf(c) >= 0)
      return "NUMERAL ";
    if("abcdefghijklmnopqrstuvwxyz".indexOf(c) >= 0)
      return "LETTER ";
    /* TODO finish implementation of operator classification
    if("+-*/&|!=".indexOf(c) >= 0)
      return "OPERATOR ";
    */
    return "UNKOWN ";
  }
}
```

- ☐ LETTER OPERATOR NUMERAL
- ☐ UNKOWN OPERATOR NUMERAL
- ☐ LETTER UNKOWN NUMERAL
- ☐ LETTER OPERATOR UNKOWN
- ☐ Does not compile
- ☐ Other: _____

8 Count Loops [26]

The following program counts the number of iterations of a loop and prints the count when the loop terminates. What does it print?

```
public class InTheLoop {
  public static final int END = Integer.MAX_VALUE;
  public static final int START = END - 100;
  public static void main(String[] args) {
    int count = 0;
    for (int i = START; i <= END; i++) count++;
    System.out.println(count);
  }
}
```

- ☐ 99
- ☐ 100
- ☐ 101
- ☐ Does not compile
- ☐ Other: _____

9 Never Ending Story [28, 29, 30, 32, 33]

Again you have to write some code at your own, and again you simply have to provide some declarations used within loops. Your aim is to make the loops continue forever.

```
{ // first looper
  __________ i = __________;
  for (int i = start; i <= start + 1; i++) { }
} { // second looper
  __________ i = __________;
  while (i == i + 1) { }
} { // third looper
  __________ i = __________;
  while (i != i) { }
} { // fourth looper
  __________ i = __________;
  while (i != i + 0) { }
} { // fifth looper
  __________ i = __________;
  __________ j = __________;
  while (i <= j && j <= i && i != j) { }
}
```

10 Overloaded Constructors [46]

This puzzle presents you with two `Confusing` constructors. The `main` method invokes a constructor, but which one? The program's output depends on the answer. What does the program print, or is it even legal?

```
package at.spardat.puzzlers;

public class Confusing {
  public Confusing(Object o) {
    System.out.println("Object");
  }
  public Confusing(double[] d) {
    System.out.println("double array");
  }
  public static void main(String[] args) {
    new Confusing(null);
  }
}
```

- ☐ Object
- ☐ double array
- ☐ Does not compile
- ☐ Other: _____

11 Which Instance [50]

This puzzle tests your understanding of Java's two classiest operators: instanceof and cast. What does each of the following three programs do?

```
public class Type1 {
  public static void main(String[] args) {
    String s = null;
    System.out.println(s instanceof String);
  }
}
```

☐ Prints "true" ☐ Prints "false" ☐ Prints "null" ☐ Does not compile

☐ Other: _____

```
public class Type2 {
  public static void main(String[] args) {
    System.out.println(new Type2() instanceof String);
  }
}
```

☐ Prints "true" ☐ Prints "false" ☐ Prints "null" ☐ Does not compile

☐ Other: _____

```
public class Type3 {
  public static void main(String[] args) {
    Type3 t = (Type3) new Object();
  }
}
```

☐ Does not compile ☐ Other: _____

12 What's the point? [51]

```
class Point {
  private final int x, y;
  private final String name; // cached at construction time
  public Point (int x, int y) {
    this.x = x; this.y = y;
    this.name = this.makeName();
  }
  protected String makeName() {
    return "[" + x + "," + y + "]";
  }
  public final String toString() {
    return this.name;
  }
}

public class ColorPoint extends Point {
  private final String color;
  public ColorPoint(int x, int y, String color) {
    super(x, y);
    this.color = color;
  }
  protected String makeName() {
    return super.makeName() + ":" + color;
  }
  public static void main(String[] args) {
    System.out.println(new ColorPoint(4, 2, "purple"));
  }
}
```

The program given above has two immutable value classes, one class represents a point with integer coordinates and the second class adds a bit of color to the puzzle. The main porgram creates and prints an instance of the second class. What does it print?

☐ [4,2] ☐ [4,2]:purple ☐ Does not compile

☐ Other: _____

13 Null and Void [54]

Here is yet another variant on the classic “Hello World” program. What does this one do?

```
public class Null {
  public static void greet() {
    System.out.println("Hello world!");
  }
  public static void main(String[] args) {
    System.out.println(((Null) null).greet());
  }
}
```

☐ Prints "Hello World!" ☐ Throws a NullPointerException ☐ Does not compile

☐ Other: _____

14 Name It [57]

This program consists of a simple immutable class that represents a name, with a main method that puts a name into a set and checks whether the set contains the name. What does the program print?

```
public class Name() {
  private final String first, last;
  public Name(String first, String last) {
    this.first = first; this.last = last;
  }
  public boolean equals(Object o) {
    if(!(o instanceof Name)) return false;
    Name n = (Name) o;
    return n.first.equals(first) && n.last.equals(last);
  }
  public static void main(String[] args) {
    Set<Name> set = new HashSet<Name>();
    set.add(new Name("Spar", "Dat"));
    System.out.print(set.contains(new Name("Spar", "Dat")));
  }
}
```

☐ true ☐ false ☐ Does not compile

☐ Other: _____

15 Shades of Gray [68]

This program has two declarations of the same name in the same scope and no obvious way to choose between them. Does the program print Black? Does it print White? Is it legal?

```
public class ShadesOfGray {
  public static void main(String[] args) {
    System.out.println(X.Y.Z);
  }
}

class X {
  static class Y {
    static String Z = "Black";
  }
  static C Y = new C();
}

class C {
  String Z = "White";
}
```

☐ Black ☐ White ☐ Does not compile

☐ Other: _____

16 Reflection Infection

This puzzle illustrates a simple application using reflection. What does this program print?

```
public class {
  public static void main(String[] args) {
    Set<String> set = new HashSet<String>();
    set.add("foo");
    Iterator it = set.iterator();
    Method m = it.getClass().getMethod("hasNext");
    System.out.println(m.invoke(it));
  }
}
```

☐ true ☐ false ☐ Does not compile

☐ Other: _____

17 Lazy Initialization [85]

```
public class Lazy {
  private static boolean initialized = false;
  static {
    Thread thread = new Thread(new Runnable() {
      public void run() {
        initialized = true;
      }});
    thread.start();
    try {
      thread.join();
    } catch(InterruptedException e) {
      throw new AssertionError(e);
    }
  }
  public static void main(String[] args) {
    System.out.println(initialized);
  }
}
```

This poor little class is too lazy to initialize itself in the usual way, so it calls on the help of a background thread. What does the program print? Is it guaranteed to print the same things every time you run it?

☐ true ☐ false ☐ Undetermined ☐ Does not compile

☐ Other: _____

18 Class Warfare [93]

This puzzle tests your knowledge of binary compatibility: What happens to the behavior of one class when you change another class on which the first class depends? More specifically, suppose that you compile the following two classes.

```
at.spardat.puzzler.client;

public class PrintWords {
  public static void main(String[] args) {
    System.out.println(
       Words.FIRST + " " + Words.SECOND + " " + Words.THIRD);
  }
}
```

```
at.spardat.puzzler.library;

public class Words {
  private Words() { }
  public static final String FIRST  = "the";
  public static final String SECOND = null;
  public static final String THIRD  = "set";
}
```

Now suppose that you modify the library class as follows and recompile it but not the client:

```
at.spardat.puzzler.library;

public class Words {
  private Words() { }
  public static final String FIRST  = "physics";
  public static final String SECOND = "chemistry";
  public static final String THIRD  = "biology";
}
```

What does the client program print?

☐ physics chemistry biology ☐ the null set

☐ Other: _____