Java Puzzles

Michał Bejm

December 7, 2009

It's Elementary

```
public class Elementary {
  public static void main(String[] args) {
    System.out.println(12345 + 54321);
  }
}
```

What Does It Print?

(a) 12345
(b) 17777
(c) 66666
(d) Won't compile

What Does It Print?

(a) 12345
(b) 17777
(c) 66666
(d) Won't compile

Why?

It's El-ementary

```
public class Elementary {
  public static void main(String[] args) {
    System.out.println(l2345 + 5432l);
  }
}
```

It's El-ementary

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

Always use a capital el (L) in long literals, never a lowercase el (l).

```
System.out.println(12345 + 5432L);
```

Line Printer

```
public class LinePrinter {
 public static void main(String[] args) {
  //Note:\u000A is representation of newline
  char c = 0x000A;
  System.out.println(c);
 }
}
```

What Does It Print?

(a) 0x000A

(b) 10

(c) Two blank lines

(d) Won't compile

What Does It Print?

(a) 0x000A

(b) 10

(c) Two blank lines

(d) Won't compile

Why?

Explanation

```
LinePrinter.java:3:  ';' expected

//Note:\u000A is representation of newline

                     ^

1 error
```

Explanation

```
LinePrinter.java:3:  ';' expected

//Note:\u000A is representation of newline

                     ^

1 error
```

Unicode escape breaks comment in two!

This is what the parser sees

```
public class LinePrinter {
 public static void main(String[] args) {
  //Note:
is representation newline
  char c = 0x000A;
  System.out.println(c);
 }
}
```

How Do You Fix It?

```
public class LinePrinter {
 public static void main(String[] args) {
  //Escape sequences (like \n) are fine
  //in comments
  char c = '\n';
  System.out.println(c);
 }
}
```

Escape Rout

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

What Does It Print?

(a) 2

(b) 16

(c) 26

(d) Won't compile

What Does It Print?

(a) 2
(b) 16
(c) 26
(d) Won't compile

Why?

Explanation

The program prints the value of the expression "a".length() + "b".length()

Explanation

The program prints the value of the expression "a".length() + "b".length()

If the Unicode escapes in the original program are replaced with \", it will print 16:
System.out.println("a \".length() + \"b".length());

Explanation

The program prints the value of the expression "a".length() + "b".length()

If the Unicode escapes in the original program are replaced with \", it will print 16:
System.out.println("a \".length() + \"b".length());

Unicode escapes are dangerous
Use escape sequences instead, if possible

WTF?

```
\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020
\u0063\u006c\u0061\u0073\u0073\u0020\u0055\u0067\u006c\u0079
\u007b\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020
\u0020\u0020\u0020\u0020\u0073\u0074\u0061\u0074\u0069\u0063
\u0076\u006f\u0069\u0064\u0020\u006d\u0061\u0069\u006e\u0028
\u0053\u0074\u0072\u0069\u006e\u0067\u005b\u005d\u0020\u0020
\u0020\u0020\u0020\u0020\u0061\u0072\u0067\u0073\u0029\u007b
\u0053\u0079\u0073\u0074\u0065\u006d\u002e\u006f\u0075\u0074
\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u0028\u0020
\u0022\u0048\u0065\u006c\u006c\u006f\u0020\u0077\u0022\u002b
\u0022\u006f\u0072\u006c\u0064\u0022\u0029\u003b\u007d\u007d
```

WTF?

```
\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020
\u0063\u006c\u0061\u0073\u0073\u0020\u0055\u0067\u006c\u0079
\u007b\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020
\u0020\u0020\u0020\u0020\u0073\u0074\u0061\u0074\u0069\u0063
\u0076\u006f\u0069\u0064\u0020\u006d\u0061\u0069\u006e\u0028
\u0053\u0074\u0072\u0069\u006e\u0067\u005b\u005d\u0020\u0020
\u0020\u0020\u0020\u0020\u0061\u0072\u0067\u0073\u0029\u007b
\u0053\u0079\u0073\u0074\u0065\u006d\u002e\u006f\u0075\u0074
\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u0028\u0020
\u0022\u0048\u0065\u006c\u006c\u006f\u0020\u0077\u0022\u002b
\u0022\u006f\u0072\u006c\u0064\u0022\u0029\u003b\u007d\u007d
```

- Is it another Perl hacker?

WTF?

```
\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020\u0020
\u0063\u006c\u0061\u0073\u0073\u0020\u0055\u0067\u006c\u0079
\u007b\u0070\u0075\u0062\u006c\u0069\u0063\u0020\u0020\u0020
\u0020\u0020\u0020\u0020\u0073\u0074\u0061\u0074\u0069\u0063
\u0076\u006f\u0069\u0064\u0020\u006d\u0061\u0069\u006e\u0028
\u0053\u0074\u0072\u0069\u006e\u0067\u005b\u005d\u0020\u0020
\u0020\u0020\u0020\u0020\u0061\u0072\u0067\u0073\u0029\u007b
\u0053\u0079\u0073\u0074\u0065\u006d\u002e\u006f\u0075\u0074
\u002e\u0070\u0072\u0069\u006e\u0074\u006c\u006e\u0028\u0020
\u0022\u0048\u0065\u006c\u006c\u006f\u0020\u0077\u0022\u002b
\u0022\u006f\u0072\u006c\u0064\u0022\u0029\u003b\u007d\u007d
```

- Is it another Perl hacker?
- No, It's just another Java hacker

Isn't it obvious?

Here is how it looks after the Unicode escapes are translated to the characters they represent (after cleaning up the formatting):

```
public class Ugly {
    public static void main(String[] args) {
        System.out.println("Hello w" + "orld");
    }
}
```

Minute by Minute

```
public class Clock {
    public static void main(String[] args) {
        int minutes = 0;
        for (int ms = 0; ms < 60*60*1000; ms++)
            if (ms % 60*1000 == 0)
                minutes++;
        System.out.println(minutes);
    }
}
```

What Does It Print?

(a) 60
(b) 60000
(c) 3600000
(d) None of the above

What Does It Print?

(a) 60
(b) 60000
(c) 3600000
(d) None of the above

Why?

Explanation

- The problem lies in the boolean expression
  `(ms % 60*1000 == 0)`

Explanation

- The problem lies in the boolean expression
  `(ms % 60*1000 == 0)`
- Expression above is not equivalent to
  `(ms % 60000 == 0)`

Explanation

- The problem lies in the boolean expression
  `(ms % 60*1000 == 0)`
- Expression above is not equivalent to
  `(ms % 60000 == 0)`
- The `%` and `*` operators have the same precedence

Explanation

- The problem lies in the boolean expression
  `(ms % 60*1000 == 0)`
- Expression above is not equivalent to
  `(ms % 60000 == 0)`
- The `%` and `*` operators have the same precedence
- The expression `(ms % 60*1000)` is equivalent to
  `(ms % 60) * 1000`

Fixed program

```
public class Clock {
private static final int MS_PER_HOUR=60*60*1000;
private static final int MS_PER_MINUTE=60*1000;
    public static void main(String[] args) {
        int minutes = 0;
        for (int ms = 0; ms < MS_PER_HOUR; ms++)
            if (ms % MS_PER_MINUTE == 0)
                minutes++;
        System.out.println(minutes);
    }
}
```

No Pain, No Gain

```
import java.util.Random;

public class Rhymes {
 private static Random rnd = new Random();
 public static void main(String[] args) {
  StringBuffer word = null;
  switch(rnd.nextInt(2)) {
   case 0:  word = new StringBuffer('P');
   case 1:  word = new StringBuffer('G');
   default:  word = new StringBuffer('M');
  }
  word.append('a');
  word.append('i');
  word.append('n');
  System.out.println(word);
 }
}
```

What Does It Print?

(a) Pain, Gain, or Main (varies at random)

(b) Pain or Gain (varies at random)

(c) Main (always)

(d) None of the above

What Does It Print?

(a) Pain, Gain, or Main (varies at random)

(b) Pain or Main (varies at random)

(c) Main (always)

(d) None of the above: ain (always)

Why?

Another look

```
import java.util.Random;

public class Rhymes {
 private static Random rnd = new Random();
 public static void main(String[] args) {
  StringBuffer word = null;
  switch(rnd.nextInt(2)) {              // No breaks!
   case 0:  word = new StringBuffer('P');
   case 1:  word = new StringBuffer('G');
   default:  word = new StringBuffer('M');
  }
  word.append('a');
  word.append('i');
  word.append('n');
  System.out.println(word);
 }
}
```

How to fix it?

```
import java.util.Random;

public class Rhymes {
 private static Random rnd = new Random();
 public static void main(String[] args) {
  StringBuffer word = null;
  switch(rnd.nextInt(3)) {
   case 0:  word = new StringBuffer("P"); break;
   case 1:  word = new StringBuffer("G"); break;
   default:  word = new StringBuffer("M"); break;
  }
  word.append('a');
  word.append('i');
  word.append('n');
  System.out.println(word);
 }
}
```

Elegent version

```
import java.util.Random;

public class Rhymes {
  private static Random rnd = new Random();
  public static void main(String args[]) {
    System.out.println("PGM".charAt(rnd.nextInt(3))
    +"ain");
  }
}
```

Bibliography

"Java Puzzlers: Traps, Pitfalls, and Corner Cases"
By Joshua Bloch and Neal Gafter

Thanks for your attention