Answer Key — Week 13 Parsons Problems

For TAs only — do not distribute

Problem 1 — Wardrobe

```
public class Wardrobe {
  private String[] items;

  public Wardrobe(String[] items) { this.items = items; }

  public String checkInventory(int slot) {
    try {
      return "Slot contains: " + items[slot];
    } catch (ArrayIndexOutOfBoundsException e) { return "Slot is empty!"; }
  }
}
```

DISTRACTOR `} catch (NullPointerException e) { return "Slot is empty!"; }`

Wrong exception type — `items` is always set by the constructor, so it is never `null`; indexing it with an out-of-range `slot` throws an `ArrayIndexOutOfBoundsException` instead, which this catch clause does not match. It propagates uncaught and crashes the program with a stack trace.

DISTRACTOR `public Wardrobe(String[] items) { items = items; }`

Missing `this.` — without it, `items = items;` just reassigns the constructor's own parameter to itself, leaving the field `items` `null`. Every call to `checkInventory` then throws a `NullPointerException` when indexing into it, which — same as the distractor above — the `ArrayIndexOutOfBoundsException` catch clause does not match.

Not part of the slips — `WardrobeDemo.java` is provided as-is to test the assembled solution:

```
public class WardrobeDemo {
  public static void main(String[] args) {
    Wardrobe wardrobe = new Wardrobe(new String[] {"scarf", "jacket"});
    System.out.println(wardrobe.checkInventory(Integer.parseInt(args[0])));
  }
}
```

Problem 2 — Calculator

```
public class Calculator {
    static int divideEvenly(int a, int b) throws DivisorException {
        if (b == 0) throw new DivisorException("Cannot divide by zero!");
        if (a % b != 0) throw new DivisorException(a + " is not evenly divisible by " + b +
"!");
        return a / b;
    }
}

class DivisorException extends Exception {
    DivisorException(String message) { super(message); }
}
```

DISTRACTOR `DivisorException(String message) { }`

Missing the `super(message)` call — without it, the message is never passed to the `Exception` superclass, so `e.getMessage()` returns `null` instead of `Cannot divide by zero!`.

DISTRACTOR `static int divideEvenly(int a, int b) throws ArithmeticException {`

Wrong exception declared — `DivisorException` is a **checked** exception, so any method that throws it must declare exactly that in its `throws` clause. Declaring `throws ArithmeticException` instead does not satisfy this, so it does not compile: `unreported exception DivisorException; must be caught or declared to be thrown`.

Not part of the slips — `CalculatorDemo.java` is provided as-is to test the assembled solution:

```
public class CalculatorDemo {
    public static void main(String[] args) {
        try {
            System.out.println("Result: " + Calculator.divideEvenly(100,
Integer.parseInt(args[0])));
        } catch (DivisorException e) {
            System.out.println(e.getMessage());
        }
    }
}
```