Question 1 (20 marks total, 2 marks for each part):

Part A:
Suppose that the `Counter` class that we studied in the course had the following implementation:

```
public class Counter {
    private static int value;

    public Counter(int v) {
        if (v < 0) {
            throw new IllegalArgumentException();
        }
        value = v;
    }

    public advance() {
        value++;
    }
}
```

Consider the following fragment of code:

```
Counter c1 = new Counter(5);
Counter c2 = new Counter(10);
c2.advance();
```

What is the value of the counter `c1`?

Part B:
What object does `this` refer to inside of a non-static method?

Part C:
What is the definition of the term *class invariant*?

Part D:
Provide a complete Javadoc documentation comment for the following method:

```
public static double divide(double a, double b) {
    if (b == 0.0) {
        throw new ArithmeticException("division by zero");
    }
    return a / b;
}
```

Part E:
Consider the `Period` class that was discussed during a lecture (and in the notebooks). Why is the `Period` class a composition of two `Date` objects instead of being an aggregation of two `Date` objects?

Part F:
Java's wrapper class `Double` has the following class declaration:

```
public final class Double extends Number implements Comparable<Double>
```

The `Number` class has the following class declaration:

```
public abstract class Number implements Serializable
```

List all the types that a `Double` object has.

Part G:
Consider the following line of Java code:

```
Number n = new Double(1.0);
```

What is the declared type of `n`?
What is the runtime type of `n`?

Part H:
`Double` is a direct subclass of `Number`. What should the first line of the `Double` constructor do?

Part I:
In terms of computational complexity, what advantage does a linked node-based stack have over an array-based stack?.

Part J:
Consider the `ArrayIterator` class that implements an iterator for our array-based list implementation. The `hasNext` method that we studied returns true if there are one or more elements in the list that are *after* the iterator's current position. Provide an implementation of the method `hasPrevious` that returns true if there are one or more elements in the list that are *before* the iterator's current position.