Contracts and Exceptions

Programmer roles

- Relative to a given module a programmer can be:
  - Producer, when participant in its development
  - Consumer, when using the module

Contracts: documentation and verification

```
/**
 * Returns the double closest to the square root of its argument
 * value.
 *
 * @pre 0 ≤ value
 * @post result >= 0
 * @param value the value whose square root will be approximated
 * @throws IllegalArgumentException If value < 0
 */
static public double squareRoot(final double value) {
    if(value < 0)
        throw new IllegalArgumentException("Message");
    …
    assert … : "Informative message";
    return result;
}
```

Asserts, by default, are not active. To activate execute the JVM with option -ea.

Invariant

```
public Rational(final int numerator,
                final int denominator) {
    if(denominator == 0)
        throw new IllegalArgumentException("Message");

    this.numerator = numerator;
    this.denominator = denominator;
    normalize();

    checkInvariant();
}


private void checkInvariant() {
    assert 0 < denominator : "…";
    assert gcd(numerator, denominator) == 1 : "…";
}

…
```

Where? Final instruction of constructors , starting inspectors, (starting?) and ending modifiers.

The invariant must hold for an object of the class for the instance to be considered coherent.

Private methods

```java
private void reduce() {
    assert denominator != 0;

    if (denominator < 0) {
        denominator = -denominator;
        numerator = -numerator;
    }

    final int gcd = gcd(numerator, denominator);
    numerator /= gcd;
    denominator /= gcd;

    assert gcd(numerator, denominator) == 1 : "…";
    assert 0 < denominator : "…";
}
```

Pre-condition of the method.

Not necessary to verify invariant.

Summary

- Contracts and Exceptions