# Testen

## Lösungen zu den Übungsaufgaben

# Übung: Objektzähler



```java
/**
 * Die Klasse Element zählt ihre Instanzen zur Laufzeit und gibt den aktuellen
 * Zählerstand im Konstruktor aus.
 *
 * @author abo781
 *
 */
public class Element {

  /**
   * Statischer Zähler (Objektübergreifend)
   */
  private static int zaehler;

  /**
   * Konstruktor. Inkrementierung des Zählers und Ausgabe.
   */
  public Element() {
    zaehler++;
    System.out.println("Anzahl Elemente: " + zaehler);
  }

  /**
   * Programmeinstieg.
   */
  public static void main(String[] args) {
    new Element();
    new Element();
    new Element();
  }
}
```

# Übung: Black- vs. Whitebox Test



- Blackbox-Test
  - eigentlich genauer Spezifikation erforderlich
  - z.B. gültiger Wertebereich (z.B. positive, ganze Zahlen > 0)
  - dann Extremfälle und repräsentative Beispiele wählen
  - Soll mit Ist abgleichen
  - ggt(1,1) = 1
  - ggt(1,1000) = 1
  - ggt(1000,1) = 1
  - ggt(1000,1000) = 1000
  - ...
- Whitebox-Test
  - Quellcode Zeile für Zeile analysieren
  - sicherstellen, dass jede Zeile bei mindestens einem Test durchlaufen wird
  - Extremfälle für jede Anweisung mit je mind. einem Testfall abdecken

# Übung: JUnit-Test



```java
/**
 * Testklasse für die Klasse Bruch (hier nur für die Methode
berechneGgt.
 */
public class TestBruch {
  @Test
  public void testBerechneGgt() {
    // Standardfälle
    assertEquals("12,24 -> 1", 1, Bruch.berechneGgt(23, 42));
    assertEquals("12,24 -> 1", 12, Bruch.berechneGgt(12, 24));
    assertEquals("16,12 -> 4", 4, Bruch.berechneGgt(16, 12));
    // Randfälle
    assertEquals("1,1000 -> 1", 1, Bruch.berechneGgt(1, 1000));
    assertEquals("1000,1 -> 1", 1, Bruch.berechneGgt(1000, 1));
    assertEquals("1,1 -> 1", 1, Bruch.berechneGgt(1, 1));
    assertEquals("1000,1000 -> 1000", 1000, Bruch.berechneGgt(1000,
1000));
  }
}
```

# Übung: Fehlertypen



```
int y = 23.42;
```

→ statischer Semantikfehler, int, double nicht kompatibel

```
for ( int i = 0; i < 5; i--){ ... }
```

→ Logikfehler: Endlosschleife

```
int z = 3 / ( 4 - 4);
```

→ dynamischer Semantikfehler: Division durch 0

```
int x = 23
```

→ Syntaxfehler: fehlendes Semikolon

# Übung: Platzhalterobjekte



```java
/**
 * Stub-Klasse für eine Webseite. Clou: Die Klasse merkt sich den gesendeten
 * Text und kann ihn wieder zurückgeben. Damit können wir testen.
 */
public class WebseiteStub {

  /**
   * Hier wird der letzte Text gespeichert.
   */
  private String letzerText = "";

  /**
   * Dummy-Methode zum Darstellen des Texts.
   */
  public void stelleDar(String text) {
    letzerText = text;
  }

  /**
   * Getter.
   */
  public String getLetzterText() {
    return letzerText;
  }

}
```

```java
/**
 * Testklasse für WebseiteAusgabe.
 */
public class WebAusgabeTest {

  @Test
  public void testGibAus() {
    WebseiteStub ausgabeStub = new WebseiteStub();
    WebAusgabe ausgabe = new WebAusgabe(ausgabeStub);
    String testText = "Die ist der Text-Text";
    ausgabe.gibAus(testText);
    assertEquals("Texte passen nicht.",
ausgabeStub.getLetzterText(), testText);
  }
}
```