

# Advanced Aspects of Object-Oriented Programming (SS 2009)

## Practice Sheet 1

Date of Issue: 21.04.09  
Deadline: 27.04.09  
(until 10 a.m. as PDF via E-Mail)

### Exercise 1 Lecture concepts

Explain the following lecture concepts by giving reasonable examples:

- encapsulation & information hiding
- polymorphism & subtyping
- declarative & model-based specifications
- functional & non-functional properties

### Exercise 2 Java Programming (1)

Fix the following program such that it prints either the word Pain, Gain or Main, randomly.

---

```
import java.util.*;

public class Rhymes {
    private static Random rnd = new Random();

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

---

Listing 1: Rhymes.java

### Exercise 3 Identity vs Equality of Objects

- a) Use the Java Language Specification (<http://java.sun.com/docs/books/jls/download/langspec-3.0.pdf>) to explain the output of the following program

---

```
public class IdentityEquality
{
    public static void main(String ... args)
    {
        String a1="A";
        String a2="A";

        String b1=new String("B");
        String b2=new String("B");

        System.out.println("a1==a2:␣"+(a1==a2));
        System.out.println("b1==b2:␣"+(b1==b2));

        System.out.println("a1.equals(a2):␣"+(a1.equals(a2)));
        System.out.println("b1.equals(b2):␣"+(b1.equals(b2)));
    }
}
```

---

Listing 2: IdentityEquality.java

- b) Modify the previous example such that an identity comparison is sufficient to test for equality. *Note: Have a look at the Java API documentation of the `String` class.*
- c) Which property of the `String` class is necessary for the previous solution.

## Exercise 4 The equals Method

The JDK description of the equals-Method is as follows:

The equals method implements an equivalence relation on non-null object references:

- It is reflexive: for any non-null reference value `x`, `x.equals(x)` should return true.
- It is symmetric: for any non-null reference values `x` and `y`, `x.equals(y)` should return true if and only if `y.equals(x)` returns true.
- It is transitive: for any non-null reference values `x`, `y`, and `z`, if `x.equals(y)` returns true and `y.equals(z)` returns true, then `x.equals(z)` should return true.
- It is consistent: for any non-null reference values `x` and `y`, multiple invocations of `x.equals(y)` consistently return true or consistently return false, provided no information used in equals comparisons on the objects is modified.
- For any non-null reference value `x`, `x.equals(null)` should return false.

The equals method for class `Object` implements the most discriminating possible equivalence relation on objects; that is, for any non-null reference values `x` and `y`, this method returns true if and only if `x` and `y` refer to the same object (`x == y` has the value true).

Note that it is generally necessary to override the `hashCode` method whenever this method is overridden, so as to maintain the general contract for the `hashCode` method, which states that equal objects must have equal hash codes.

- a) Analyze the following code fragment checking for the aforementioned properties.

---

```
public class Date {
    private int y,m,d;

    public Date(int y, int m, int d) {
        this.y = y;
        this.m = m;
        this.d = d;
    }

    public boolean equals(Object obj) {
        if (obj instanceof Date) {
            Date date = (Date)obj;
            return date.y == y && date.m == m && date.d == d;
        }
        return false;
    }
}
```

---

Listing 3: Date.java

---

```
public class NamedDate extends Date {
    private String name;

    public NamedDate(int y, int m, int d, String name) {
        super(y,m,d);
        this.name = name;
    }

    public boolean equals(Object other) {
        if (other instanceof NamedDate && !name.equals(((NamedDate)other).name))
            return false;
        return super.equals(other);
    }
}
```

---

Listing 4: NamedDate.java