University of Zurich^UZH

s.e.a.l. software evolution & architecture lab

Informatics I – EProg HS15

Exercise 2

1 Task: Primitive and Reference Types

1.1 Learning Objectives

1. Be able to explain the difference between primitive types and reference types.
2. Practice assignment operations with primitive types and reference types.

1.2 Assignment

a) Data Type Theory

Compare primitive types with reference types by completing table 1. You can use keywords.

	Primitive Type	Reference Type
Programming Language Integration		provided by the Java Standard Library (e.g.[1] String, Integer) or extensible with user-defined class definitions
Size in Memory		variable
Memory Layout (What is stored?)	the actual value	
Can have methods? (yes/no)		
Can store other data? (yes/no)		
Default Value	predefined but basically just zero (e.g. 0 for `int`, false for `boolean`)	

Table 1: Data Type Comparison

	Primitive Type	Reference Type
Programming Language Integration	built-in data type that is part of the language definition and does not come from a class definition	provided by the Java Standard Library (e.g.[2] String, Integer) or extensible with user-defined class definitions
Size in Memory	fixed	variable
Memory Layout (What is stored?)	the actual value	a reference to an object
Can have methods? (yes/no)	no	yes
Can store other data? (yes/no)	no	yes
Default Value	predefined but basically just zero (e.g. 0 for int, false for boolean)	null

Table 2: Solution: Data Type Comparison

Solution:

b) Functioning

What is the output of the following code snippets? Explain your answer!

```java
int alpha = 5;
int beta = 2;

alpha = beta;
alpha++;

System.out.println(alpha);
System.out.println(beta);
```

Listing 1: Snippet 1

Solution:

Output: 3 and 2
Motivation: Assignments with primitive data types *copy the value* from the right-hand side (RHS) of the assignment operator into the variable on the left-hand side (LHS).
Line 4: `alpha` takes the value 2 of `beta`.
Line 5: `alpha` (currently 2) is then incremented by 1 to 3.
`beta` always remains 2.

Assume that the class `Number` can save an integer number via `setNumber(int newNumber)` and output its current value to the console via `printNumber()`. What is the output of the following code snippets? Explain your answer!

```
 1  Number x = new Number();
 2  Number y = new Number();
 3  x.setNumber(6);
 4  y.setNumber(8);
 5
 6  x = y;
 7
 8  x.printNumber();
 9  y.printNumber();
10
11  x.setNumber(10);
12  y.setNumber(2);
13
14  x.printNumber();
15  y.printNumber();
```

Listing 2: Snippet 2

Solution:

Output:
8 and 8
2 and 2

Motivation: Assignments with reference data types *only copy the reference* from the RHS of the assignment operator into the variable on the LHS. Subsequently, both variables refer to the same object. It doesn't matter whether the object is manipulated via the variable `x` or `y`.

Line 6: From now on, variable `x` points to the same object as variable `y`. The object with the content 6 is no more referenced and therefore garbage collected by the Java Virtual Machine.
Line 8: Prints the content 8 of the object via variable `x`.
Line 9: Prints the content 8 of the object via variable `y`.
Line 11: Sets the content of the object to 10 via `x`.
Line 12: Sets the content of the object to 2 via `y`. We just overwrite the object's content (was 10 before) with 2.
Line 14: Prints the content 2 of the object via variable `x`.
Line 15: Prints the content 2 of the object via variable `y`.

2 Task: Statements

2.1 Learning Objectives

1. Know elementary kinds of statements.

2.2 Assignment

Provide two different examples for each kind of statement from below:

a) Declaration

Solution:

```
String name;
int i;
```

b) Initialization

Reuse the variables declared in a).

Solution:

```
name = "Mike";
i = 7;
```

c) Send a Message

Solution:

```
name.toUpperCase();
System.out.println("Test");
```

d) Combined Declaration and Initialization

Solution:

```
String name = "Mike";
int i = 7;
```

3 Task: Code Comprehension

3.1 Learning Objectives

1. Learn how to understand a class and its behavior by using the Java API with the example of the `String` class.
2. Practice reading and understanding code.

3.2 Assignment

Which of the following code snippets are syntactically correct and what is the output in that case? Explain your answer!

You can use:

- Walter Savitch, Java: An Introduction to Problem Solving and Programming
- Java API

```
String s;
s = "Java is great!";
System.out.println(s);
```

Listing 3: Snippet 1

Solution:

Correct.
Output: `Java is great!`

```
String s;
"Java is great!" = s;
System.out.println(s);
```

Listing 4: Snippet 2

Solution:

Wrong.
Output: Does not compile.
Motivation: There must be always a variable on the left-hand side (LHS) of the assignment operator. You can never assign a variable to a value! It's always vice versa.

```
String s = "Welcome";
s.toUpperCase();
System.out.println(s);
```

Listing 5: Snippet 3

Solution:

Correct.
Output: `Welcome`
Motivation: In general, String objects are immutable, i.e. they cannot be manipulated. Therefore, the method `toUpperCase()` of the String object returns a new, manipulated String object. If you want that line 3 outputs `WELCOME`, you need to store this new object at line 2 with `s = s.toUpperCase();`.

```java
System.out.println("hELLo".toUpperCase());
```

Listing 6: Snippet 4

Solution:

Correct.
Output: `HELLO`
Motivation: The new String object returned by `toUpperCase()` is passed to the `println()` method. The output is the same as if the result would have been stored in a variable like this:
`String s = "hELLo".toUpperCase()`
`System.out.println(s)`

```java
String s = "Welcome";
PrintStream t = new PrintStream("at university");
t = s;
System.out.println(t);
```

Listing 7: Snippet 5

Solution:

Wrong.
Output: Does not compile.
Motivation: „at university“ is no object of the class `PrintStream`. You cannot assign the „Welcome“ `String` to a variable of type `PrintStream`.

```java
String s = System.out.println("Welcome");
```

Listing 8: Snippet 6

Solution:

Wrong.
Output: Does not compile.
Motivation: `println()` does not return a `String`. Actually, `println()` returns nothing.

```java
String s = new "Hello";
System.out.println(s);
```

Listing 9: Snippet 7

Solution:

Wrong.
Output: Does not compile.
Motivation: Create objects from classes with `new <class name>()`. The class name (e.g. String) is missing.

```java
String s = "h" + "ell".toUpperCase() + "o";
System.out.println(s);
```

Listing 10: Snippet 8

Solution:

Correct.
Output: `hELLo`

```java
String s = new String("hello".toUpperCase);
System.out.println(s);
```

Listing 11: Snippet 9

Solution:

Wrong.
Output: Does not compile.
Motivation: Wrong syntax when calling `toUpperCase()`. The brackets are missing.

```java
String s;
String t = "Welcome";
s = "at university";
t = s;
s = t;
System.out.println(s);
System.out.println(t);
```

Listing 12: Snippet 10

Solution:

Correct.
Output: `at university` and `at university`
Motivation: After line 4, t points to the same object as s. This object has the content „at university“. Additionally, the object with the content „Welcome“ is lost since we don’t have a reference anymore.
Line 5 is useless since t and s point to the same object and therefore doesn’t change anything.

```java
String s = new String("welcome").toUpperCase();
String t = "welcome".toUpperCase();
System.out.println(s);
System.out.println(t);
```

Listing 13: Snippet 11

Solution:

Correct.
Output: `WELCOME` and `WELCOME`
Motivation: Although the output is the same, s and t are different. (s points to a wrapper String class allocated in the heap whereas t points to an optimized String pool maintained by the JVM. You’ll later see that comparing s and t with `s == t` will return false.)
Conclusion: Never initialize Strings with the `new` operator!

4 Task: Cascading and Composition

4.1 Learning Objectives

1. Learn and practice cascading and composition of method calls with the example of String operations.

4.2 Assignment

Given the following declarations and initializations:

```
String s1 = "butterfly";
String s2 = "tiger";
```

a) Determine Result

What is the result of the following method calls?

1. `"gold".substring(1, 4).concat(s1.substring(6,7)).concat(s2.substring(2));`

   Solution:
   oldfger

2. `s1.substring(2).concat(s1.substring(2,3)).concat(s1.substring(1,3));`

   Solution:
   tterflytut

b) Create Call Chain

Create call chains like above that produce the results below. You must only use the variables `s1`, `s2` and methods `substring()`, `concat()`. Note that multiple solutions are possible.

1. flyger

   Solution:
   `s1.substring(6).concat(s2.substring(2));`

2. buttiger

   Solution:
   `s1.substring(0,3).concat(s2);`

5 Task: Object-oriented Concepts

5.1 Learning Objectives

1. Know the difference between classes and objects.
2. Understand the common terms of object-oriented concepts.

5.2 Assignment

1. Decide for each term of Table 3 which OO-concept (object-oriented-concept) is most appropriate. Is it a class, an object, a method, or an attribute?

Term	Class	Object	Method	Attribute
cat				
the cat "Garfield"				
color of the cat fur				
best friend of the cat				
the car "Herbie"				
car				
car number				
accelerate				
Porsche car				

Table 3: OO-Terms

Solution:

Term	Class	Object	Method	Attribute
cat	x			
the cat "Garfield"		x		
color of the cat fur				x
best friend of the cat		x		
the car "Herbie"		x		
car	x			
car number				x
accelerate			x	
Porsche car	x			

Table 4: Solution: OO-Terms

2. Explain the following terms in the context of object-oriented programming:
   (a) Class

Solution:

A class is a kind of blueprint for a family of objects. All objects of the same class have similar data and the same methods.

(b) Instance

Solution:

A specific example of a class (German: Eine konkrete Ausprägung einer Klasse). An object is always an instance of a class in Java.

(c) Message

Solution:

Mechanism that allows to call a method. It consists of the method name followed by a list of arguments. Note that the list of arguments may also be empty (i.e. a method with no arguments)

(d) Reference

Solution:

A link (memory address) that points to a concrete object in the heap.

(e) Overloading

Solution:

The definition of multiple methods in a class with an identical name but different (although typically similar) behavior. These methods must have a different signature which means that the number/or type of their arguments must be different.

Example: Imagine the following methods in the same class:

```
.convertToString(int number)
.convertToString(double number)
```

They have the same name (`convertToString`) but different argument types (`int` and `double`).

6 Task: Classes

6.1 Learning Objectives

1. You can implement a class that encapsulates state via attributes and behavior via methods.
2. You can verify the behavior of the implemented class with a TestDriver.

6.2 Assignment

a) Multiplier

Write a class with the name `Multiplier` and the methods `reset()`, `multiply()`, and `result()`.

- The default value of the result is 1.
- The method `multiply()` takes a single parameter of type `long` and multiplies the parameter by the result stored within the object.
- `result()` prints the result to the screen.
- `reset()` sets back the object by resetting the result to 1.

```java
public class Multiplier {

   . . .

   public void multiply( . . . ) {
      . . .
   }

   public void reset() {
      . . .
   }

   public void result() {
      . . .
   }
}
```

Solution:

```java
/**
 * Multiplies given numbers to a stored result.
 */
public class Multiplier {

   private long result = 1;

   /**
    * Multiplies the given number by the result.
    */
   public void multiply(long number) {
      result = result * number;
   }

   /**
    * Resets the result.
    */
   public void reset() {
      result = 1;
   }

   /**
    * Prints the result to the console.
    */
   public void result() {
      System.out.println(result);
   }
}
```

Listing 14: `Multiplier` Class

b) MultiplierTestDriver

Write a class called `MultiplierTestDriver` with a `main()` method wherein you create at least 3 `Multiplier` class instances. Test the methods `reset()`, `multiply()`, and `result()`. Also test the assignment operator (=) and call methods afterward (e.g. something like `multiplier1 = multiplier2` followed by `adder1.result()`).

Solution:

```java
/**
 * Tests a few Multiplier objects on their method functionality.
 */
public class MultiplierTestDriver {

   public static void main(String[] args) {

      Multiplier m1 = new Multiplier();
      Multiplier m2, m3;
      m2 = new Multiplier();
      m3 = new Multiplier();

      m1.multiply(2);
      m1.multiply(4);
      m1.result(); // Output: 8

      m2.multiply(3);
      m2.multiply(7);
      m2.multiply(2);
      m2.result(); // Output: 42

      m1.reset();
      m1.result(); // Output: 1
      m2.result(); // Output: 42

      m1.multiply(1);
      m2.multiply(2);
      m3.multiply(3);

      m3 = m2;
      m3.result(); // Output: 84
      m2.result(); // Output: 84

      m2.multiply(2);
      m3.multiply(2);
      m2.result(); // Output: 336
      m3.result(); // Output: 336

   }
}
```

Listing 15: `MultiplierTestDriver` Class