Software Development I – Exercises **Winter Term 2012/2013**

Übungen zu Softwareentwicklung 1 **Assignment 6**

Name: ____________________	Teaching Assistant: ____________________
Student ID (Matr.Nr.): ____________________	Points (max. 24): ____________________
Group: ____________________	Deadline: Wed, Dec. 5th, 2012 12:00 noon
Instructor: ____________________	Editing time: ____________________

Question 1: Smartphone Shop — 8 points

Given a fictitious shop (class `Shop`) selling smartphones, write a program that can manage heterogeneous devices with different technical features. Each phone (class `Phone`) has a name, a "unique" product number, and five more specifications (CPU, storage, resolution, OS, weight). All the attributes should be stored as strings (data type `String`).

The class `Phone` should provide a constructor with the phone's name as parameter (data type `String`). Furthermore, you are requested to initialize the product number with a 10-digit(!) random integer number (you can start with `Math.random()` to get a pseudo random number in the interval [0;1[).
The class `Shop` stores the phones in an array of phones (`Phone[]`) of an arbitrary but explicitly set length (use a constant for creating the array – see the example below). Test your program by filling your array with a number of phones (at least 3) and printing them to the screen. Your console output should look similar to this:

```
Phone 1: Apple iPhone 5 [#2716346936]
        -CPU: 1.3 GHz dual core Apple A6
        -OS: iOS 6.0.1
        -Resolution: 640x1136
        -Storage: 64 GB
        -Weight: 112 g
Phone 2: Samsung Galaxy S III [#3087884400]
        -CPU: 1.4 GHz quad-core Cortex-A9
        -OS: Android 4.0 (Ice Cream Sandwich)
        -Resolution: 720x1280
        -Storage: 32 GB
        -Weight: 133 g
```

Use *setter* methods to set the phone's features and *getter* methods for the output. Have a look at the following Java code snippet to get an idea how to solve this exercise.

```java
// Initialize the shop
Shop shop = new Shop();

// Initialize the array as container for numerous smartphones.
// Use a constant for the array length
Phone[] phones = new Phone[MAX_NUM_PHONES];

// Create a new phone and set its features and attributes
Phone apple = newPhone("Apple iPhone 5");
apple.setCpu("1.3 GHz dual core Apple A6");
// ...

// Add the phone to the shop's phone array
phones[0] = apple;

// Add and print the phones to the screen
shop.addPhones();
shop.printPhones();
```

Test your program by creating and adding some phones in a private method `addPhones()` which you can call in the `main` method of the class `Shop` (see example above). You do not need to implement a user dialog, just fill the array by using the constructor and the setter methods of the class `Phone` (as can be seen in the code example above). Provide the source code for all implemented classes.

Write the program in Java. Provide all the material requested below at the very bottom ("Requested material…")

Question 2: Interactive Electronic Shop **16 points**

Extend the basic functionality of your code from question 1 to give the user a more interactive experience when visiting your shop. Furthermore, your shop can now manage arbitrary electronic devices (e.g., smartphones, laptops, tablets, MP3 player, etc.) which are implemented as products (class `Product`).

Class **`Product`**: Besides the mandatory fields (name, price, 10-digit product number), a product can have a maximum number of 10 optional features (class `Feature`) which are stored in a feature array (`Feature[]`). A feature consists of a name and a description both implemented as strings.

Class **`Shop`**: The electronic shop has a name and an array of products (`Product[]`) to store a maximum number of 100 devices (use a constant to initialize the array).

To make the implementation a bit easier for you, stick to the given interface and implement at least the following methods (in addition to the general *getter* and *setter* methods that you will need):

Shop:

```
// The constructor
public Shop(String name)  {...}

// Method to add a product p to the shop
public boolean addProduct(Product p) {...}

// Method to remove a product at a certain index
public boolean removeProduct(int index) {...}

// Get a product at a certain index
public Product getProduct(int index) {...}

// Returns a string representation of all the products
public String printProducts() {...}
```

Product:

```
// The constructor
public Product(String name, double price)  {...}

// Method to add a feature f to the product
public boolean addFeature(Feature f) {...}

// Method to remove a feature at a certain index
public boolean removeFeature(int index) {...}

// Returns a string representation of all the features
public String printFeatures() {...}
```

Implement a simple interactive dialog on the console where the user can always pick one option by entering a single number. The following examples show, how the user interaction could look like:

Printing the product list:

```
========================================
  My TestShop
========================================

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
1
[0] PC (#9828286677)
[1] Phone (#6633696897)
[2] Tablet (#9743925129)
[3] MP3-Player (#9844343263)

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
```

Adding a product to the shop:

```
========================================
  My TestShop
========================================

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
2
Please enter the product's name: NewName
Please enter the product's price: 99.99
Please enter a feature (name=description) or hit enter to return: Name1=Desc1
Please enter a feature (name=description) or hit enter to return: Name2=Desc2
Please enter a feature (name=description) or hit enter to return: Name3
Invalid input. Please try again.
Please enter a feature (name=description) or hit enter to return: Name3=Desc3
Please enter a feature (name=description) or hit enter to return:

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
1
[0] PC (#7743008378)
[1] Phone (#8172776236)
[2] Tablet (#4805652465)
[3] MP3-Player (#7847288797)
[4] NewName (#3320841722)
```

Deleting a product from the shop:

```
========================================
  My TestShop
========================================

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
3
[0] PC (#3917852306)
[1] Phone (#2278893349)
[2] Tablet (#2137225131)
[3] MP3-Player (#1316591533)

Please select the index of the product to delete: 2
```

```
Product removed successfully!

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
1
[0] PC (#3917852306)
[1] Phone (#2278893349)
[2] MP3-Player (#1316591533)
```

Changing the name of a product:

```
========================================
  My TestShop
========================================

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
4
[0] PC (#2645449902)
[1] Phone (#8915678527)
[2] Tablet (#4192119749)
[3] MP3-Player (#9466447194)

Please select the index of the product:
2
Product Tablet (#4192119749) selected: [Price=549.0], [Brand=ASUS], [USB Ports=1],
[Resolution=1280x800 pixels], [Weight=635 g]

Please select option:
[1] Change product name
[2] Change product price
[3] Change product features
[4] Return
1
Current name is: "Tablet"
Enter new name: TabletNew
Product name changed successfully!

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
1
[0] PC (#2645449902)
[1] Phone (#8915678527)
[2] TabletNew (#4192119749)
[3] MP3-Player (#9466447194)
```

Modifying the features of a product:

```
========================================
  My TestShop
========================================

Please select option:
[1] Print product list
[2] Add product
[3] Delete product
[4] Show product details
[5] Exit program
4
[0] PC (#6863665806)
[1] Phone (#5761482749)
[2] Tablet (#4618101920)
[3] MP3-Player (#7138583416)

Please select the index of the product:
2
```

```
Product Tablet (#4618101920) selected: [Price=549.0], [Brand=ASUS], [USB Ports=1],
[Resolution=1280x800 pixels], [Weight=635 g]

Please select option:
[1] Change product name
[2] Change product price
[3] Change product features
[4] Return
3
The selected product has the following features:
[0] [Brand=ASUS]
[1] [USB Ports=1]
[2] [Resolution=1280x800 pixels]
[3] [Weight=635 g]

Please select one option:
[1] Delete an existing feature
[2] Add a new feature
[3] Return
1
[0] [Brand=ASUS]
[1] [USB Ports=1]
[2] [Resolution=1280x800 pixels]
[3] [Weight=635 g]

Please enter the index of the feature to be deleted: 3
Feature successfully removed.

Please select one option:
[1] Delete an existing feature
[2] Add a new feature
[3] Return
1
[0] [Brand=ASUS]
[1] [USB Ports=1]
[2] [Resolution=1280x800 pixels]
```

Hints:

- Splitting the user dialog into several methods (e.g., `printAddProductDialog()`, `printDeleteProductDialog()`, `printProductDetailsDialog()`, etc.) might be useful.
- Use the method `String[] java.lang.String.split(String regex)` to split the feature entered by the user into the fields `'name'` and `'description'`.

Write the program in Java. Provide all the material requested below at the very bottom ("Requested material…")

Requested material for all programming problems:

- **For each exercise, hand in the following:**
  - **a) The idea for your solution written in text form.**
  - **b) Source code (Java classes including English(!) comments).**
  - **c) Test plan for analyzing boundary values (e.g., min. temperature allowed, maximum number of input, etc.) and exceptional cases (e.g., textual input when a number is required). State the expected behavior of the program for each input and make sure there is no "undefined" behavior leading to runtime exceptions. List all your test cases in a table (test case #, description, user input, expected (return) values).**
  - **d) The output of your java program for all test cases in your test plan.**
- Pay attention to using adequate and reasonable data types for your implementation, check the user input carefully and print out meaningful error messages.

Advice for user input: While no longer mandatory it is still recommended to use the class Input.java for user input (read operations) in your program.