Assignment 2

Instructor: Musard Balliu

October, 2015

In this assignment, you will write an object-oriented application for a shopping cart software.The software allows shopping customers to accumulate a list of products for purchase. Upon checkout, the software should calculate the total price for a order and show the products to the customer.

The task

In this assignment, you are given the specifications of a shopping cart application which you need to implement. The application consists of the following classes:

- A `Product` class representing a product to be purchased
- A `DiscountedProduct` subclass representing a product with $x\%$ discount applied to it
- A `Buy2Take3Product` subclass representing a product for which one can buy 2 pieces and get a third for free
- A `Cart` class representing the products purchased by a given customer
- A `Main` class representing the interface between the customer and the shopping cart application

The specification takes the form of a set of commented java files, included in the following sections and available for download from
http://www.cse.chalmers.se/∼musard/dit948/assign2_java.zip.

The interface, example session

The class `Main` has a `main` method which, when executed, should lead to an interactive user session of the following kind:

```
Welcome to DIT958 shop
What's your name?
Jib
Hi Jib. Please choose one of the following options:

Enter  1 to buy a product
Enter  0 to checkout and proceed with the payement
1
Product name: >CD of Leonard Cohen
Seller: Javier
Price: 30
How many: 4
Discount (enter 0 if no discount applies): 0
Does Buy2Take3 apply? Y/N Y
Enter  1 to buy a product
Enter  0 to checkout and proceed with the payement
1
Product name: TV
Seller: Maria
Price: 5000
How many: 1
Discount (enter 0 if no discount applies): 20
Does Buy2Take3 apply? Y/N N
Enter  1 to buy a product
Enter  0 to checkout and proceed with the payement
0
CD of Leonard Cohen    22.50 SEK. Sold by Javier
CD of Leonard Cohen    22.50 SEK. Sold by Javier
CD of Leonard Cohen    22.50 SEK. Sold by Javier
CD of Leonard Cohen    22.50 SEK. Sold by Javier
TV [discounted by 20.00%]    4000.00 SEK. Sold by Maria

In total you have to pay 4090.0 SEK
```

Additional test cases are available in the file TestCases.txt.

The classes

Class `Main`

```
/**
 *  Class representing a shopping cart application for the
 *  second assignment in DIT948, 2015 edition.
 *  This is the main class for the application, interacting
 *  with the customer, adding to the cart different products to
 *  be purchased and finally calculating the total amount to be payed
 */

public class Main {

/**
 *  Allows a shopkeeper to enter the details for a product
 *  to be purchased by a customer
 *  @param cart the shopping cart of a given customer
 */
 public static void askCustomer(Cart cart){
 // Code to read from console the product name, seller,
 // price, number of products, discount and
 // if Buy2Take3 applies.

// Then create a product of the correct type
// and add it to the shopping cart
 }

// Main method to interact with a customer
public static void main(String[] args) {
 println("Welcome to DIT958 shop");
 println("What's your name?");
 String customer = readLine();
 println("Hi "+customer+". Please choose one of the following options:");
 println("");

 Cart cart = new Cart();

 //Implement the user interface here
 // Ask the user to choose 0 (buy product) or
 // 1 (checkout), depending on  what they want to do.
 // Use the method askCustomer
 // See TestCases.txt for several examples

}
}
```

Class Cart

```
/**
 *  Class representing a shopping cart for the
 *  second assignment in DIT948, 2015 edition.
 */

public class Cart  {

// array of products (max 100 products)
private final Product[] products = new Product[100];

//position of the first free cell to add a product
private int position =0;

/**
 * Return the list of products
 * @return
 */
public Product[] getProducts(){
       // code here
}

/**
 * Add a product to the list
 * @param product
 */
public void addProduct(Product product) {
       // code here
}

/**
 * Add an array of products to the list
 * @param products
 */
public void addProducts(Product[] products) {
       // code here
}

/**
 * Adds a product several times
 * @param product
 * @param howManyTimes number of times to add product
 */
public void addProduct(Product product, int howManyTimes) {
       // code here
}

/**
 * Calculate the total price
 * @return
 */
public double totalPrice(){
       // code here
}

/**
 * String representation of products in a shopping cart
 * Example:
 * CD of Leonard Cohen 22.50 SEK. Sold by Javier
 * TV [discounted by 20.00%]    4000.00 SEK. Sold by Maria
 */
public String toString() {
String result = "";
       // code here
return result;
}
}
```

Class Product

```
/**
 *  Class representing a product to be purchased for the
 *  second assignment in DIT948, 2015 edition.
 */
public class Product {
// Name of the seller
private final String seller;

// Name of the product
private final String name;

// Price of the product
private final double price;

/**
 * Construct a new Product given the following parameters
 * @param seller
 * @param name
 * @param price
 */
public Product(String seller, String name, double price) {
}

/**
 * Construct a new Product from a given product
 * @param original
 */
public Product(Product original) {
// Use the constructor implemented above
}

/**
 * Return the seller of  this product
 * @return seller
 */
public final String getSeller() {
//code here
}

/**
 * Return the name of  this product
 * @return name
 */
public final String getName() {
       // code here
}

/**
 * Return the price of  this product
 * @param cart
 * @return price
 */
public double getPrice(Cart cart) {
       // code here
}

/**
 * Returns true if the price of this product
 * can be reduced
 * @return
 */
public boolean canBeReduced() {
//Nothing to do here
return true;
}

/**
 * Return the name of the product
 */
public String toString() {
       // code here
}
}
```

Class DiscountedProduct

```
/**
 *  Subclass representing a  discounted product to be
 *  purchased for the second assignment in DIT948, 2015 edition.
 *  It extends the <tt>Product<tt> class with two instance
 *  variables
 */
```

```
public class DiscountedProduct extends Product {

// Original product
private final Product original;

// Discount in percent (%)
private final double discount;

/**
 * Construct a discounted product
 * @param original
 * @param discount
 */
public DiscountedProduct(Product original, double discount) {
       // if the price can not be reduced you should print a message and
       // terminate the program. Use "System.exit(0);" to terminate.
       // code here

}

/**
 * Return the price of this discounted product
 * @param cart shopping cart
 * @return discounted price
 */
public double getPrice(Cart cart) {
       // code here
}

/**
 * Return the string representation of the product
 * Example: CD [discounted 20 %]
 */
public String toString() {
       // code here
}
}
```

Class Buy2Take3Product

```
/**
 *  Subclass representing a  product to be  purchased
 *  (using the formula "buy 2 take 3") for the second
 *  assignment in DIT948, 2015 edition.
 *  It extends the <tt>Product<tt> class with one instance
 *  variable
 */

public class Buy2Take3Product extends Product {

//original product
private final Product original;

/**
 * Construct a Buy2Take3 product
 * If the price of this product can not be reduced
 * you should print a message to the user and terminate the
 * program
 * @param original
 */
public Buy2Take3Product(Product original) {
       // if the price can not be reduced you should print a message and
       // terminate the program. Use "System.exit(0);" to terminate.
       // code here
}

/**
 * Return false if the product price can not be
 * reduced
 * @return
 */
public boolean canBeReduced() {
// You can not discount the price of Buy2Take3 product
// code here
}

/**
 * Return the unit price of a product using the
 * formula "Buy2Take3"
 * @param cart shopping cart
 * @return unit price
 */
public double getPrice(Cart cart) {

// calculate unit price of this product purchased
// as Buy2Take3
// code here
}
}
```

Remarks and Notation

You can assume that a user will never buy more than 100 products

If a user buys the same product twice, these instances should be considered as buying different products (see last Test 4 in Testcases.txt)

The program should terminate if the user chooses both the discount and the offer Buy2Take3, since Buy2Take3 products can not be discounted.

Method $getPrice$ requires $cart$ as parameter, since in the subclasses the price of a product may depend on other products in the cart. Think what happens for objects of type $Buy2Take3$.

You may use the method String.format(...) to format the strings as in TextCases.txt.

You may use either the `dit948` input-output or the standard Java libraries.

Grading

- To get a G you can assume that the user never applies one offer for another offer as part of the input. Namely, a Buy2Take3 product is never discounted, and vice versa.
- To get a VG it is necessary implement all tasks in the assignment.

  Note that these requirements are necessary, but not sufficient, to get the respective grades (see Administrative matters).

Administrative matters

Strive for readable code with appropriate comments! While the ultimate test of a program is that it does what it's supposed to do, **I should be able to read the program and understand it**.

The assignment is to be completed in groups of one or two students. (Groups of different size should first be agreed with the course instructor.)

It must be uploaded to the GUL system **by 23:55 on October 14**.

Please note: **The submission is to be made via GUL. It's no good sending it to me, either before or after the deadline!**

Only java source files should be uploaded, no class files. Your java files should be put in a zip-archive with the following name:

```
assign2_author1_author2.zip
```

where author1, author2 are the surnames (family names) of the group members. You must also supply a README-file (README.txt) containing the names of the authors and their social security numbers, together with a brief statement about each author's contribution. For example, "author1 has been responsible for user input and author2 for the program logic". A statement such as "All authors have contributed equally to all aspects of the program" is also acceptable.

Note that **each author must submit individually via the GUL** (even though it is the same program!). Only students who submit via the GUL can be graded. All comments etc. will be posted on GUL.

After the deadline has passed, each group **must present solutions to the TAs during the first Thursday supervision session**. Exceptions are to be agreed with the TAs. Group members are expected to know and explain all parts of the code despite their statement of contribution.