**Home exercise 5**

**Course name: Object oriented programming and design for engineering**

**Course number: 157109**

**Subject: Generics and Java Streams**

# Part A – Theoretical questions

This part contains several questions on Java Generics. The answers should be submitted in a pdf file.

## Question 1

Given the following interface, revise it so that *consume* accepts a generic type.

**public** **interface** **Consumer** {

**public** **void** **consume**(String parameter);

}

## Question 2

Below you can find two code snippets:

Snippet 1:

**List** list = **new** **ArrayList**();

list.add(1);

**Object** o = list.get(0);

**String** str = (String) o;

Snippet 2:

**List**<String> list = **new** **ArrayList** ();

list.add(1);

**Object** o = list.get(0);

**String** str = (String) o;

Which of them will cause a compilation error and which a runtime error. Explain why.

## Question 3

Consider the following function

public static <T> void f(List<T> argument) {…}

Is it possible to determine in runtime what is the type parameter of the argument? Explain why.

## Question 4

Given the following generic function:

**class Program {**

**public** **static** <T> T **returnType**(T argument) { **return** argument; }

}

Will the following code pass compilation? Will it run correctly? If not explain what should be changed to make it work.

**Integer** inferredInteger = **Program**.returnType(1);

**String** inferredString = **Program**.returnType("String");

## Question 5

You are given the following class:

**public** **class** **Farm** {

**private** List<Animal> animals = new ArrayList<Animal>();

**public** **void** **addAnimals**(Collection<Animal> newAnimals) {

animals.addAll(newAnimals);

}

}

The Animal class is extended by the Cat and Dog classes. After creating an instance *farm* of type Farm, the following code caused compilation errors:

List<Cat> cats = new ArrayList<Cat>();

List<Dog> dogs = new ArrayList<Dog>();

Farm farm = new Farm();

farm.addAnimals(cats); // Compilation error

farm.addAnimals(dogs); // Compilation error

Explain why there are compilation errors. Can you change the code of Farm, so that the given code will run correctly?

# Part B – Programming

**An important requirement**: In this exercise you must not use loops or stream’s foreach function. Any repetitive operation should be implemented using streams.

The information about products, customers and orders of a computing products store is kept in a dedicated information system.



* Each customer has several (zero or more) orders
* Each order has several products
* Each product can belong to several orders

The data management and processing define a data layer, a business layer, and a user interface. The user interface is provided in the Program class (It should not be modified)

## The data layer

The data layer is composed of four data objects:

* Customer
  + id – The customer ID number
  + name – First and last name
  + tier – A field that can have the value 1, 2, or 3. Represents how profitable is the customer for the store. Tier 3 is the most profitable.
* Product
  + code – Product code
  + name – Product name
  + category – The possible categories are PC, Printers, Displays, Storage, Network, Cameras, Gaming, Software, Accessories
  + price – product price
* Order
  + orderId – The order ID number
  + orderDate – The date the order was created
  + deliveryDate – The date the order was delivered
  + status – The possible statuses are AwaitingPayment, Processing, Pickup, Complete, Cancelled
  + customerID – The ID of the customer who made the order
* OrderProduct
  + orderId - The order ID number
  + code – Product code
  + quantity – the number of items of the product that were ordered

Each class will have a constructor that accepts a String, which describes the object details. This String will be saved in line in a corresponding file. You are provided with the store’s data files.

**Requirement 1:** Implement the constructors that initialize the object’s fields according to the information provided in the string.

In order to manage the store’s data, we will maintain four static lists, defined in the DataSource class:

* Customers
* Products
* Orders
* Order-Products

**Requirement 2:** Fill the lists with information based on the provided files:

* customers.txt
* products.txt
* orders.txt
* orderProducts.txt

Note: Use streams to read the file’s contents

## Business layer:

The business layer is represented by the IBL interface, which is the contract for implementing this layer (design by contract). BL class will implement IBL.

The following describes the IBL interface:

Product getProductById(long productId); //Returns a product by its ID (code)

Order getOrderById(long orderId); //Returns an order by its ID

Customer getCustomerById(long customerId); //Returns a customer by its ID

List<Product> getProducts(ProductCategory cat, double price); // Returns the list of products that belong to category *cat* with price equal or smaller than *price*. The list should be sorted by the product ID.

List<Customer> popularCustomers(); // Returns the list of popular customers. A customer is popular if it belongs to tier 3 and it has more than 10 orders. The list should be sorted by the customer ID.

List<Order> getCustomerOrders(long customerId); // Returns the list of orders made by the customer. The list should be sorted by the order ID.

long numberOfProductInOrder(long orderId); // Returns the number of unique products in an order.

List<Product> getPopularOrderedProduct(int orderedQuantity);// Returnes the list of products, for which the total ordered quantity is at least orderedQuantity (hint: use summingInt). The list should be sorted by the product ID.

List<Product> getOrderProducts(long orderId); // Returns the list of products that are in a specific order. The list should be sorted by the product ID.

Product getMaxOrderedProduct(); //Returns the product that was ordered the most times (Meaning that it appears in many orders, disregarding quantity)

List<Customer> getCustomersWhoOrderedProduct (long productId); // Returns the number of customers who ordered the product. The list should be sorted by the customer ID.

double sumOfOrder(long orderID); // Returns the total cost of a given order

List<Order> getExpensiveOrders(double price); //Returns the list of orders that have a total cost larger than *price*. The list should be sorted by the order ID.

List<Customer> ThreeTierCustomerWithMaxOrders(); // Returns the list of tier 3 customers who have the maximal number of orders. The list should be sorted by the customer ID.

Running example

The following defines the required output for the provided data files and Program class.

---------getProductById:--------

Product: 495 product495 category: Displays price: 113.0

---------getOrderById:--------

order: 241 order date: 23/01/2022 delivery date: 25/01/2022 status: Processing customr id: 41

---------getCustomerById:--------

customer: 284 name: customer284 tier: 3

---------getProducts:--------

Product: 43 product43 category: Accessories price: 32.0

Product: 87 product87 category: Accessories price: 25.0

Product: 167 product167 category: Accessories price: 39.0

Product: 246 product246 category: Accessories price: 35.0

Product: 321 product321 category: Accessories price: 47.0

Product: 403 product403 category: Accessories price: 38.0

Product: 431 product431 category: Accessories price: 43.0

---------popularCustomers:--------

customer: 1 name: customer1 tier: 3

customer: 7 name: customer7 tier: 3

customer: 93 name: customer93 tier: 3

customer: 97 name: customer97 tier: 3

---------getCustomerOrders:--------

order: 83 order date: 08/01/2022 delivery date: 30/01/2022 status: Processing customr id: 83

order: 183 order date: 05/01/2022 delivery date: 29/01/2022 status: Cancelled customr id: 83

order: 283 order date: 19/01/2022 delivery date: 25/01/2022 status: Pickup customr id: 83

order: 383 order date: 20/01/2022 delivery date: 26/01/2022 status: AwaitingPayment customr id: 83

order: 483 order date: 20/01/2022 delivery date: 27/01/2022 status: Processing customr id: 83

order: 583 order date: 17/01/2022 delivery date: 24/01/2022 status: Pickup customr id: 83

order: 683 order date: 13/01/2022 delivery date: 02/02/2022 status: Processing customr id: 83

order: 783 order date: 03/01/2022 delivery date: 01/02/2022 status: Cancelled customr id: 83

order: 883 order date: 09/01/2022 delivery date: 30/01/2022 status: AwaitingPayment customr id: 83

order: 983 order date: 06/01/2022 delivery date: 23/01/2022 status: Pickup customr id: 83

---------numberOfProductInOrder:--------

6

---------getPopularOrderedProduct:--------

Product: 10 product10 category: Software price: 228.0

Product: 14 product14 category: Displays price: 176.0

Product: 29 product29 category: Cameras price: 295.0

Product: 36 product36 category: Accessories price: 82.0

Product: 52 product52 category: Displays price: 285.0

Product: 64 product64 category: Cameras price: 96.0

Product: 88 product88 category: Gaming price: 233.0

Product: 90 product90 category: Gaming price: 92.0

Product: 124 product124 category: Cameras price: 257.0

Product: 125 product125 category: Accessories price: 235.0

Product: 126 product126 category: Printers price: 156.0

Product: 194 product194 category: Storage price: 246.0

Product: 202 product202 category: Software price: 51.0

Product: 212 product212 category: Cameras price: 53.0

Product: 213 product213 category: Printers price: 300.0

Product: 223 product223 category: Displays price: 39.0

Product: 227 product227 category: Network price: 245.0

Product: 238 product238 category: Printers price: 169.0

Product: 276 product276 category: Accessories price: 144.0

Product: 301 product301 category: Cameras price: 138.0

Product: 332 product332 category: Displays price: 235.0

Product: 335 product335 category: Accessories price: 77.0

Product: 356 product356 category: Software price: 44.0

Product: 379 product379 category: Accessories price: 310.0

---------getOrderProducts:--------

Product: 60 product60 category: Printers price: 79.0

Product: 158 product158 category: Network price: 82.0

Product: 183 product183 category: Network price: 124.0

Product: 242 product242 category: Accessories price: 53.0

Product: 256 product256 category: Storage price: 203.0

Product: 265 product265 category: Displays price: 215.0

Product: 448 product448 category: Displays price: 152.0

---------getCustomersWhoOrderedProduct:--------

customer: 12 name: customer12 tier: 2

customer: 24 name: customer24 tier: 3

customer: 29 name: customer29 tier: 1

customer: 30 name: customer30 tier: 1

customer: 32 name: customer32 tier: 2

customer: 33 name: customer33 tier: 2

customer: 35 name: customer35 tier: 1

customer: 37 name: customer37 tier: 1

customer: 41 name: customer41 tier: 2

customer: 45 name: customer45 tier: 3

customer: 60 name: customer60 tier: 3

customer: 89 name: customer89 tier: 2

customer: 91 name: customer91 tier: 1

customer: 92 name: customer92 tier: 2

customer: 98 name: customer98 tier: 2

---------getMaxOrderedProduct:--------

Product: 36 product36 category: Accessories price: 82.0

---------sumOfOrder:--------

2396.0

---------getExpensiveOrders:--------

order: 432 order date: 05/01/2022 delivery date: 25/01/2022 status: Pickup customr id: 32

order: 521 order date: 21/01/2022 delivery date: 30/01/2022 status: Processing customr id: 21

order: 708 order date: 15/01/2022 delivery date: 30/01/2022 status: Cancelled customr id: 8

order: 800 order date: 09/01/2022 delivery date: 02/02/2022 status: Complete customr id: 0

---------ThreeTierCustomerWithMaxOrders:--------

customer: 1 name: customer1 tier: 3

customer: 7 name: customer7 tier: 3

customer: 97 name: customer97 tier: 3