IPT – Intellectual Products & Technologies Ltd.
Bl. 44- Office 2 Slatinska Str., Sofia, Bulgaria
☏ (+359-2) 973-38-38 🖷 (+359-2) 971-28-50 ✉ office@iproduct.org 🖰 http://www.iproduct.org/

PROGRAMMING EXERCISES

Problem 1: Program for invoicing

a) Write a program for invoicing, working in text mode:

b) Create a class **Product** with attributes: 1) unique identifier of the product; 2) goods code (up to 10 letters and digits); 3) name of the goods; and 4) price of goods; 4) unit of measure (kg, pcs., etc.), and methods: 1) constructor without arguments (default constructor); 2) constructor with four arguments (code, name, price, and measure of the goods); 3) method **toString()**, which returns information about the goods in text; 4) method **input()**, which to input the attributes of the goods from the keyboard. Create also a static method **main**, that creates three different products (goods), the third is entered from the keyboard and these products are printed on the screen in text.

c) Create a class **Client** with attributes: 1) identification number; 2) name; 3) address; 4) individual or company; 5) telephone (optional); 6) e-mail (optional); 6) VAT number (optional – if company is VAT Registered). Create following methods: 1) constructor without arguments (default constructor); 2) constructor with 4 arguments – mandatory attributes; 3) constructor with all arguments (full constructor); 4) method **toString()**, which returns information about Client in text, 5) method **input()**, which to input the attributes of new Client from the keyboard. Create also a static method **main**, which creates three different Clients, the third is entered from the keyboard and the data for these contractors are printed on the screen in text.

d) Create a class **Issuer** with attributes: 1) identification number; 2) name; 3) address; 4) IBAN; 5) BIC; 6) telephone; 7) VAT number (optional – if company is VAT Registered). Create following methods: 1) constructor without arguments (default constructor); 2) constructor with 6 arguments – mandatory attributes; 3) constructor with all arguments (full constructor); 4) method **toString()**, which returns information about the invoice Issuer in text, 5) method **input()**, which to input the attributes of new Issuer from the keyboard. Create also a static method **main**, which creates three different Issuers, the third is entered from the keyboard and the data for these contractors are printed on the screen in text.

e) Create a class **Invoice** with attributes: 1) unique invoice number; 2) date of issue; 3) Issuer ID number; 4) Client ID number; 5) date of payment or performance of the transaction; 6) person issuing the invoice; and 7) ordered list of one or more positions. Each position (class **InvoiceLine**) has: 1) unique identifier of the product, 2) quantity, 3) price (optional – if different from the regular price of the product). Implement following methods: 1) constructor without arguments (default constructor), 2) constructor with four arguments (number, issuer, client, person issuing the invoice, list of products), 3) constructor with all arguments (full constructor); 4) method **toString()**, which returns information about the Invoice in text. Create also a static method **main**, to create a new invoice and print it on the screen in text.

f) Create a class **InvoiceRegister** with attributes you decide and basic methods:

1. ***initialize()*** - initialize the system by creating a list of three Products and three Clients (maybe from the same subsections 1 and 2) initialize the data of the **company issuing the Invoices** and the **initial invoice number** (numbers should be auto-incremented);
2. ***createInvoice(...)*** which creates new Invoice using provided **Client ID number**; **InvoiceLines list** and and adds it to the **InvoiceRegister** and returns the **invoice number** as a result;
3. ***printInvoice(...)***, which prints the invoice with given invoice number on screen, including information about the **Issuer** and Client, number and date of invoice, products list, with displayed the name of the product, quantity, price, unit and value of each **InvoiceLine** (quantity multiplied by the price of the product), and the total amount, VAT and amount including VAT.
4. ***main(...)*** – static method that initializes the **InvoiceRegister,** creates and prints an invoice with three **InvoiceLines** (quantities are 1, 5 and 10) on the console.

Problem 2: Invoicing - separate packages

a) Move classes **Product, Client, Issuer, InvoiceLine** and **Invoice** in package **invoicing.model**

b) Move class **InvoiceRegister** in package **invoicing.controller**

c) Set the appropriate ***import*** structures and specifiers to access various classes and their methods.

Problem 3: Program to issue invoices - factoring in inheritance

a) Refactor the program from Problem 2, so that the common attributes of **Client** and **Issuer** classes are factored in a parent class called **Contragent** that is extended by **Client** and **Issuer** classes. 3) implement *polymorphic* methods **toString()** and **input()**. Modify class **InvoiceRegister** to use the new **Contragent** class. Modify also static method **main** method to test the new refactoring.

b) Refactor the program so that it is now possible to sell not only **Products**, but also **Services**. Modify the static **main** method to demonstrate the new functionality.

Literature and Internet Resources

1. Oracle® Java™ Technologies webpage – http://www.oracle.com/technetwork/java/index.html
2. Eckel, B., Thinking in Java. 4-th ed., Prentice Hall, 2006 – http://mindview.net/Books/TIJ4
3. Effective Java Second Edition, Bloch, J., Sun Microsystems, 2008
4. Schildt, H., Java 2 - Developer Guide. Softpress, in bg, 2007
5. Eck, D., Introduction to Programming Using Java, Fifth Edition, Version 5.1, June 2009 – http://math.hws.edu/javanotes/