Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем

и технологий»

**Отчёт**

по лабораторной работе №21-28

по дисциплине ***«Объектно-ориентированные технологии***

***программирования и стандарты проектирования»***

тема: «Концепции ООП, Unit testing, JCF, Input-Output stream, Serialization»

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент группы 10701119  Быков Алексей Сергеевич |
| Преподаватель: |  | Кузьмин Д. В. |

2020-2021 учебный год

**ЛАБОРАТОРНАЯ РАБОТА #21-28**

Концепции ООП, Unit testing, JCF, Input-Output stream, Serialization

**Цель работы**

Получить практические и теоретические знания в ООП.

**Tребования**

1) Скорректировать UML-диаграмму классов и интерфейсов, составляющих архитектуру основного приложения.

2) При проектировании и реализации программы рекомендуется использовать архитектурный шаблон MVC, а также фундаментальные SOLID и GRASP принципы.

3) Классы и другие сущности программы должны быть грамотно структурированы по соответствующим пакетам и иметь отражающую их функциональность названия.

4) Программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом. Работа с консолью должна быть минимальной.

5) Приложение должно корректно обрабатывать любые исключительные ситуации, которые могут возникнуть в процессе работы программы.

6) Для подтверждения работоспособности и адекватности программы, вся модель проекта должна быть покрыта соответствующими модульными тестами. Для модульного (unit) тестирования рекомендуется использовать тестовый фреймворк jUnit или TestNG.

7) Необходимо по максимуму пытаться разрабатывать универсальный код.

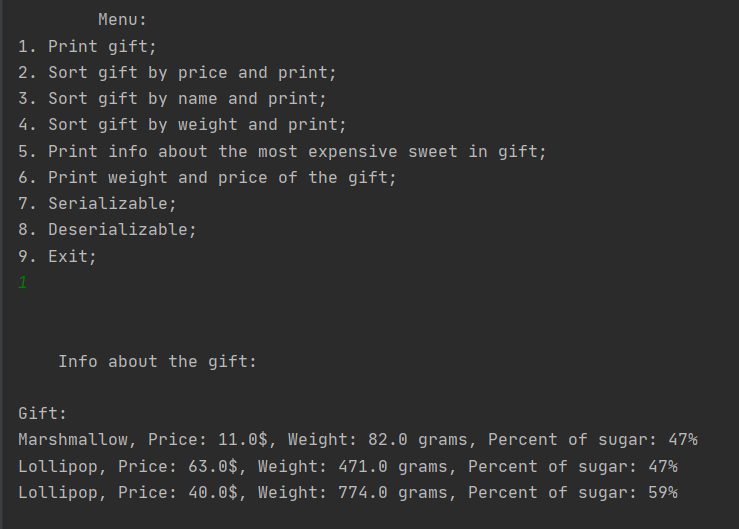
8) Программа должна обязательно быть снабжена комментариями на английском языке, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, номер группы и дату разработки.

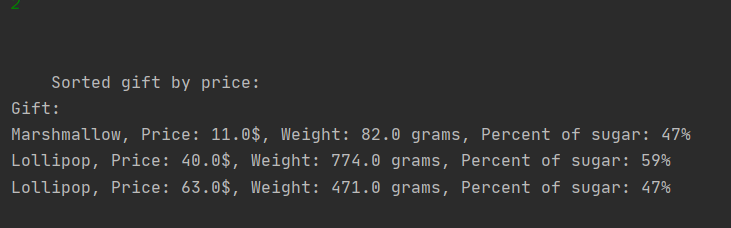
9) Исходный текст классов и демонстрационной программы рекомендуется также снабжать комментариями.

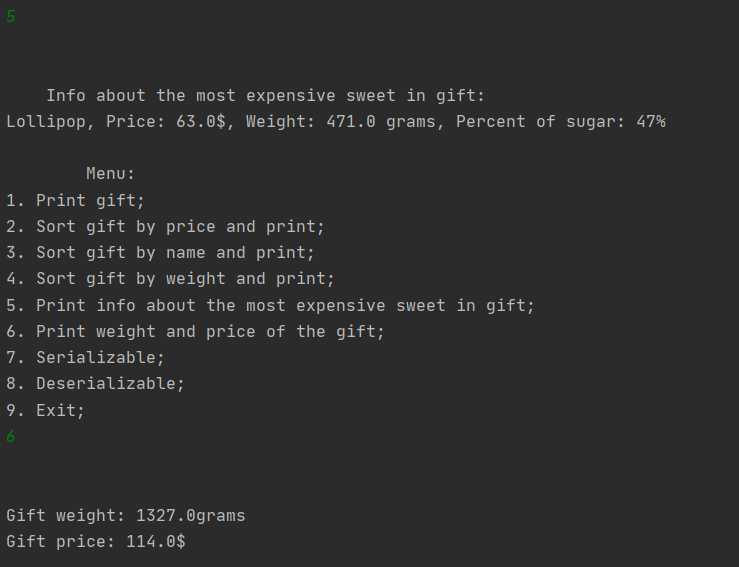
10)При разработке программ придерживайтесь соглашений по написанию кода на JAVA (Java Code-Convention).

**Задание:**

Новогодний подарок (New Year gift). Есть новогодний сладкий подарок, который состоит из конфет и прочих сладостей. Необходимо подсчитать общий вес и стоимость подарка, а также самую дорогую (недорогую) сладость







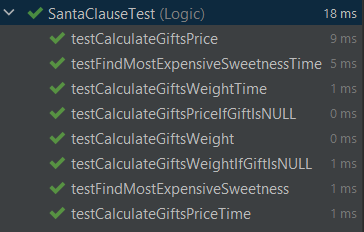


Рис. 1 – результат выполнения программы.

**Приложение А.**

**Листинг класса Main.java**

package Controller;  
  
import Model.Entity.Gift;  
import Model.Entity.Sweet;  
import Model.Exceprion.ImpossibleAmountOfSweets;  
import Model.Logic.SantaClause;  
import Util.Serializable;  
import Util.SweetsCreator;  
import Util.Initializator;  
import View.Printer;  
import java.io.IOException;  
import java.util.InputMismatchException;  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
  
  
 public static void main(String[] args) {  
 try {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Enter amount of sweets: ");  
  
 int sweetsAmount = in.nextInt();  
 List<Sweet> sweets = SweetsCreator.*create*(sweetsAmount);  
 Initializator.*init*(sweets);  
 Gift gift = new Gift(sweets);  
 int count = 0;  
 String menu = "\n\t\tMenu: " +  
 "\n1. Print gift;" +  
 "\n2. Sort gift by price and print;" +  
 "\n3. Sort gift by name and print;" +  
 "\n4. Sort gift by weight and print;" +  
 "\n5. Print info about the most expensive sweet in gift;" +  
 "\n6. Print weight and price of the gift;" +  
 "\n7. Serializable;" +  
 "\n8. Deserializable;" +  
 "\n9. Exit;\n";  
 String msg="";  
 boolean flag = true;  
 while(flag) {  
  
 //Printer.printInFile(menu);  
 Printer.*print*(menu);  
 int choice = in.nextInt();  
  
 switch (choice){  
 case 1:  
 msg = "\n\n\tInfo about the gift:\n " + gift;  
 Printer.*printInFile*(msg,count);  
 Printer.*print*(msg);  
 count++;  
 break;  
 case 2:  
 SantaClause.*sortedByPrice*(gift);  
 msg = "\n\n\tSorted gift by price:" + gift;  
 Printer.*printInFile*(msg,count);  
 Printer.*print*(msg);  
 count++;  
 break;  
 case 3:  
 SantaClause.*sortedByName*(gift);  
 msg = "\n\n\tSorted gift by name:" + gift;  
 Printer.*printInFile*(msg,count);  
 Printer.*print*(msg);  
 count++;  
 break;  
 case 4:  
 SantaClause.*sortedByWeight*(gift);  
 msg = "\n\n\tSorted gift by weight:" + gift;  
 Printer.*printInFile*(msg,count);  
 Printer.*print*(msg);  
 count++;  
 break;  
 case 5:  
 msg = "\n\n\tInfo about the most expensive sweet in gift: \n" +  
 SantaClause.*findMostExpensiveSweetness*(gift)+"\n";  
 Printer.*printInFile*(msg,count);  
 Printer.*print*(msg);  
 count++;  
 break;  
 case 6:  
 msg = "\n\nGift weight: " + SantaClause.*calculateGiftsWeight*(gift) + "grams" +  
 "\nGift price: " + SantaClause.*calculateGiftsPrice*(gift) + "$\n";  
 Printer.*printInFile*(msg,count);  
 Printer.*print*(msg);  
 count++;  
 break;  
 case 7:  
 Serializable.*serializable*(sweets);  
 count++;  
 break;  
 case 8:  
 //sweets.clear();  
 Serializable.*deserializable*(sweets);  
 count++;  
 break;  
 case 9:  
 flag = false;  
 break;  
  
 default:  
 Printer.*print*("Enter incorrect action number!!!");  
  
  
 }  
  
 }  
  
 } catch (ImpossibleAmountOfSweets ex) {  
 Printer.*print*("Input incorrect amount of sweets!!" +  
 "\nCorrect range " + SweetsCreator.*MIN\_AMOUNT* + "..." + SweetsCreator.*MAX\_AMOUNT*);  
 } catch (InputMismatchException exception) {  
 Printer.*print*("Input incorrect data!!");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**Листинг класса Gift.java**

package Model.Entity;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Gift {  
  
 public static final int *MAX\_SWEETS\_AMOUNT* = 10;  
  
 private double weight; // in grams  
 private double price; // in dollars  
 private List<Sweet> sweets;  
  
 public Gift() {  
 sweets = new ArrayList<Sweet>(0);  
 }  
  
 public Gift(List<Sweet> sweets) {  
 this.sweets = sweets;  
 }  
 public Gift(Gift gift){  
 sweets = gift.sweets;  
 weight = gift.weight;  
 price = gift.price;  
 }  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
// if(price > 0){  
// this.price = price;  
// }  
 this.price = price;  
 }  
  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public void setWeight(double weight) {  
  
 this.weight = weight;  
 }  
  
  
 public List<Sweet> getSweets() {  
 return sweets;  
 }  
  
 public void setSweets(List<Sweet> sweets) {  
 this.sweets = sweets;  
 }  
  
 public List<Sweet> getSweetsByIndex(int index) {  
 if (index >= 0 && index <= *MAX\_SWEETS\_AMOUNT*) {  
 return sweets.get(index);  
 }  
 return new Sweet(); // *TODO: throw exception* }  
  
 public void setSweetByIndex(int index, Sweet sweet) {  
 if (index >= 0 && index <= *MAX\_SWEETS\_AMOUNT*) {  
 sweets.set(index,sweet);  
 }  
  
 }  
  
 public int getAmountOfSweets(){  
  
 return sweets.size();  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder msg = new StringBuilder();  
 msg.append("\nGift:\n");  
 for(Sweet item : sweets)  
 msg.append(item ).append("\n");  
  
 return msg + "";  
 }  
}

**Листинг класса Sweet.java**

package Model.Entity;  
  
import java.io.Serializable;  
import java.util.\*;  
import java.util.function.IntFunction;  
  
public class Sweet implements List<Sweet>, Serializable {  
  
 private transient double weight;  
 private double price;  
 private String sweetsKind;  
  
 public Sweet() {  
  
 }  
  
 public Sweet(double weight, double price, String sweetsKind) {  
 this.price = price;  
 this.weight = weight;  
 this.sweetsKind = sweetsKind;  
 }  
  
 public Sweet(Sweet sweet){  
 price = sweet.price;  
 weight = sweet.weight;  
 sweetsKind = sweet.sweetsKind;  
 }  
  
 public double getWeight() {  
 return weight;  
 }  
  
 public void setWeight(double weight) {  
 this.weight = weight;  
 }  
  
  
 public double getPrice() {  
 return price;  
 }  
  
 public void setPrice(double price) {  
 this.price = price;  
 }  
  
 public String getSweetsKind() {  
 return sweetsKind;  
 }  
  
 public void setSweetsKind(String sweetnessKind) {  
 this.sweetsKind = sweetnessKind;  
 }  
  
 @Override  
 public String toString() {  
 return getSweetsKind()+  
 ", Price: " + getPrice() + "$" +  
 ", Weight: " + getWeight() + " grams";  
  
 }  
  
 @Override  
 public <T> T[] toArray(IntFunction<T[]> generator) {  
 return null;  
 }  
  
 @Override  
 public int size() {  
 return 0;  
 }  
  
 @Override  
 public boolean isEmpty() {  
 return false;  
 }  
  
 @Override  
 public boolean contains(Object o) {  
 return false;  
 }  
  
 @Override  
 public Iterator<Sweet> iterator() {  
 return null;  
 }  
  
 @Override  
 public Object[] toArray() {  
 return new Object[0];  
 }  
  
 @Override  
 public <T> T[] toArray(T[] ts) {  
 return null;  
 }  
  
 @Override  
 public boolean add(Sweet sweet) {  
 return false;  
 }  
  
 @Override  
 public boolean remove(Object o) {  
 return false;  
 }  
  
 @Override  
 public boolean containsAll(Collection<?> collection) {  
 return false;  
 }  
  
 @Override  
 public boolean addAll(Collection<? extends Sweet> collection) {  
 return false;  
 }  
  
 @Override  
 public boolean addAll(int i, Collection<? extends Sweet> collection) {  
 return false;  
 }  
  
 @Override  
 public boolean removeAll(Collection<?> collection) {  
 return false;  
 }  
  
 @Override  
 public boolean retainAll(Collection<?> collection) {  
 return false;  
 }  
  
 @Override  
 public void clear() {  
  
 }  
  
 @Override  
 public Sweet get(int i) {  
 return null;  
 }  
  
 @Override  
 public Sweet set(int i, Sweet sweet) {  
 return null;  
 }  
  
 @Override  
 public void add(int i, Sweet sweet) {  
  
 }  
  
 @Override  
 public Sweet remove(int i) {  
 return null;  
 }  
  
 @Override  
 public int indexOf(Object o) {  
 return 0;  
 }  
  
 @Override  
 public int lastIndexOf(Object o) {  
 return 0;  
 }  
  
 @Override  
 public ListIterator<Sweet> listIterator() {  
 return null;  
 }  
  
 @Override  
 public ListIterator<Sweet> listIterator(int i) {  
 return null;  
 }  
  
 @Override  
 public List<Sweet> subList(int i, int i1) {  
 return null;  
 }  
}

**Листинг класса SugarSweet.java**

package Model.Entity;  
  
public class SugarSweet extends Sweet {  
 //private String name;  
 private int percentOfSugar;  
  
 public SugarSweet(){  
  
 }  
  
  
 public SugarSweet(double weight, double price, String sweetsKind, int percentOfSugar){  
 super(weight, price, sweetsKind);  
 this.percentOfSugar = percentOfSugar;  
 }  
  
 public int getPercentOfSugar() {  
 return percentOfSugar;  
 }  
  
 public void setPercentOfSugar(int percentOfSugar) {  
 this.percentOfSugar = percentOfSugar;  
 }  
  
 @Override  
 public String toString() {  
 return getSweetsKind()+  
 ", Price: " + getPrice() + "$" +  
 ", Weight: " + getWeight() + " grams" +  
 ", Percent of sugar: " + getPercentOfSugar() + "%";  
  
 }  
}

**Листинг класса Chocolate.java**

package Model.Entity;  
  
public class Chocolate extends Sweet{  
 private int percentOfCocoa;  
  
 public Chocolate(){  
  
 }  
  
 public Chocolate(double weight, double price, String sweetsKind, int percentOfChocolate){  
 super(weight,price,sweetsKind);  
 this.percentOfCocoa = percentOfChocolate;  
 }  
  
 public int getPercentOfCocoa() {  
 return percentOfCocoa;  
 }  
  
 public void setPercentOfCocoa(int percentOfChocolate) {  
 this.percentOfCocoa = percentOfChocolate;  
 }  
  
 @Override  
 public String toString(){  
 return getSweetsKind()+  
 ", Price: " + getPrice() + "$" +  
 ", Weight: " + getWeight() + " grams" +  
 ", Percent of chocolate: " + getPercentOfCocoa() + "%";  
 }  
  
}