

МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота № 3**

з дисципліни “ Об'єктно-орієнтовне програмування ”

тема “Реалізація породжуючих шаблонів проектування.”

|  |  |  |
| --- | --- | --- |
| Виконав  студентка II курсу  групи КП-72  Городченко Анна Володимирівна  (*прізвище, ім’я, по батькові*)  варіант № 6 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2019

**Мета роботи**

Ознайомлення з основними характеристиками всіх породжуючих шаблонів («Прототипу», «Фабричного методу», «Абстрактної фабрики», «Одинака», «Будівельника»), запам’ятовування поширених ситуацій, коли використання цих шаблонів є доцільним, набуття вмінь та навичок реалізації шаблонів під час створення програмного коду.

**Постановка завдання**

1. За допомогою шаблона проектування допоможіть Діду Морозу

сформувати подарунковий набір з іграшок, цукерок тощо та його

розтиражувати. Всі складові набору повинні бути реалізовані як самостійні

об’єкти та скопійовані до кожного наступного повністю.

1. За допомогою шаблона «Абстрактна фабрика» реалізувати процес

віртуального збирання комп’ютерів та телевізорів у різних країнах. Кожна

збірка відрізняється деталями, які вбудовуються до прибору,

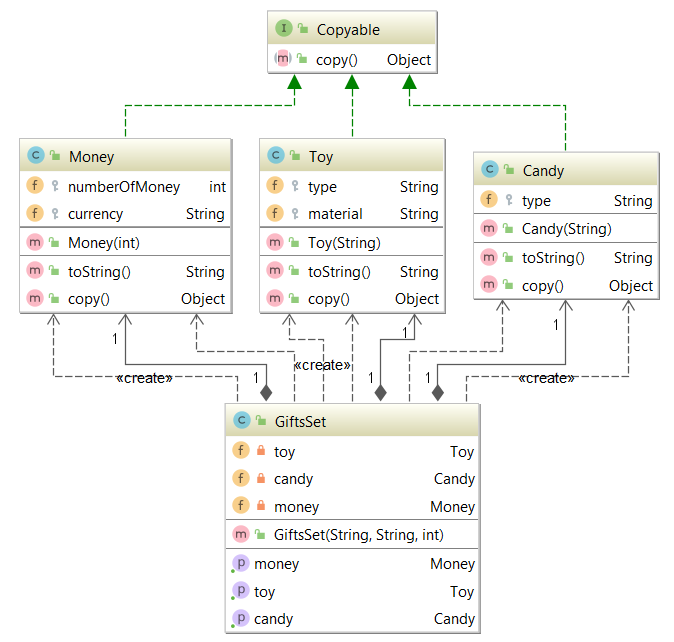
комплектацією (є базова, є vip-моделі тощо) та відповідно вартістю. В

залежності від ціни послуги забезпечити запуск виготовлення продукції у

тій чи іншій країні.

Завдання 1

Для виконання першого завдання було вирішено обрати шаблон проектування під назвою “Prototype”. Він був використаний тому, що екземпляри класу можуть знаходитися в одному зі станів, кількість яких є невеликою. Тому може бути зручніше встановити відповідну кількість прототипів та клонувати їх, а не створювати об’єкт кожного разу в потрібному стані.



**Текст коду**

|  |
| --- |
| package Prototype |
| **public interface** Copyable {  Object copy(); } |
| **public class** Candy **implements** Copyable {  **protected** String **type**;   **public** Candy(String type) {  **this**.**type** = type;  }   @Override  **public** String toString() {  **return "Candy {"** +  **"type='"** + **type** + **'\''** + **'}'**;  }   @Override  **public** Object copy() {  **return new** Candy(**type**);  } } |
| **public class** Money **implements** Copyable {  **protected int numberOfMoney**;  **protected** String **currency** = **"USD"**;   **public** Money(**int** numberOfMoney) {  **this**.**numberOfMoney** = numberOfMoney;  }   @Override  **public** String toString() {  **return "Money {"** +  **"numberOfMoney="** + **numberOfMoney** +  **", currency='"** + **currency** + **'\''** +  **'}'**;  }   @Override  **public** Object copy() {  **return new** Money(**numberOfMoney**);  } } |
| **public class** Toy **implements** Copyable {  **protected** String **type**;  **protected** String **material**;   **public** Toy(String type) {  **this**.**type** = type;  }   @Override  **public** String toString() {  **return "Toy {"** +  **"type='"** + **type** + **'\''** + **'}'**;  }   @Override  **public** Object copy() {  **return new** Toy(**type**);  }  } |
| **public class** GiftsSet {   **private** Toy **toy**;  **private** Candy **candy**;  **private** Money **money**;   **public** GiftsSet(String typeOfToy, String typeOfCandy, **int** numberOfMoney) {  **toy** = **new** Toy(typeOfToy);  **candy** = **new** Candy(typeOfCandy);  **money** = **new** Money(numberOfMoney);  }   **public** Toy getToy() {  **return** (Toy) **toy**.copy();  }   **public** Candy getCandy() {  **return** (Candy) **candy**.copy();  }   **public** Money getMoney() {  **return** (Money) **money**.copy();  }  } |
| **public class** Main {  **public static void** main(String[] args) {  GiftsSet gifts = **new** GiftsSet(**"Doll"**, **"Lolypop"**, 100);   List<Candy> candies = **new** ArrayList<>();   candies.add(gifts.getCandy());  candies.add(gifts.getCandy());  candies.add(gifts.getCandy());   **for** (Candy element : candies) {  System.***out***.println( element.toString());  }  } } |

**Приклади виконання програми**

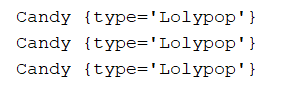
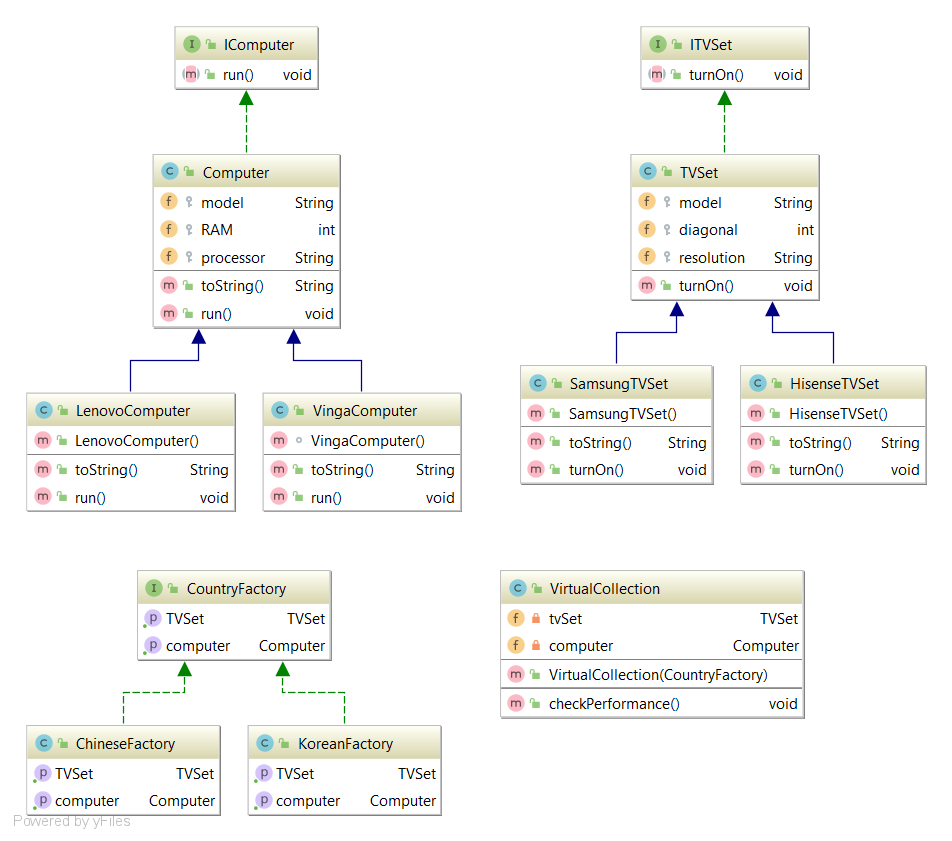


Рис. 1. Клонування цукерок з типом «Льодяник»

**Висновки**

Отже, ми використали клас, який створює об’єкти, клонуючи їх.

Завдання 2

****Для виконання другого завдання було використано шаблон проектування “Abstract factory”. Це було зроблено тому що нам треба було змінювати поведінку системи, варіюючи створювані об’єкти, але при цьому залишаючинезмінними інтерфейси.

**Текст програми**

|  |
| --- |
| interface CountryFactory |
| **public interface** CountryFactory {  TVSet getTVSet();  Computer getComputer(); } |
| class KoreanFactory |
| **public class** KoreanFactory **implements** CountryFactory {  @Override  **public** TVSet getTVSet() {  **return new** SamsungTVSet();  }   @Override  **public** Computer getComputer() {  **return new** VingaComputer();  } } |
| class ChineseFactory |
| **public class** ChineseFactory **implements** CountryFactory {    @Override  **public** TVSet getTVSet() {  **return new** HisenseTVSet();  }   @Override  **public** Computer getComputer() {  **return new** LenovoComputer();  } } |
| **Class VirtualCollection** |
| **public class** VirtualCollection {  **private** TVSet **tvSet**;  **private** Computer **computer**;   **public** VirtualCollection(CountryFactory factory){  **this**.**tvSet** = factory.getTVSet();  **this**.**computer** = factory.getComputer();  }   **public void** checkPerformance(){  **tvSet**.turnOn();  **computer**.run();  } } |
| **Class Main** |
| **public class** Main {  **public static void** main(String[] args) {   *//Chinese factory* CountryFactory china = **new** ChineseFactory();   *//Lenovo computer and Hisense TV set* Computer chineseComputer = china.getComputer();  System.***out***.println(chineseComputer.toString());   TVSet chineseTV = china.getTVSet();  System.***out***.println(chineseTV.toString());   System.***out***.println(**"\n"**);   *//Korean factory* CountryFactory korea = **new** KoreanFactory();   Computer koreanComputer = korea.getComputer();  System.***out***.println(koreanComputer.toString());   TVSet koreanTV = korea.getTVSet();  System.***out***.println(koreanTV.toString());   *//collection of specified country* VirtualCollection collection;   collection = **new** VirtualCollection(china);  System.***out***.println(**"\nChina : "**);  collection.checkPerformance();    collection = **new** VirtualCollection(korea);  System.***out***.println(**"\nKorea : "**);  collection.checkPerformance();  } } |

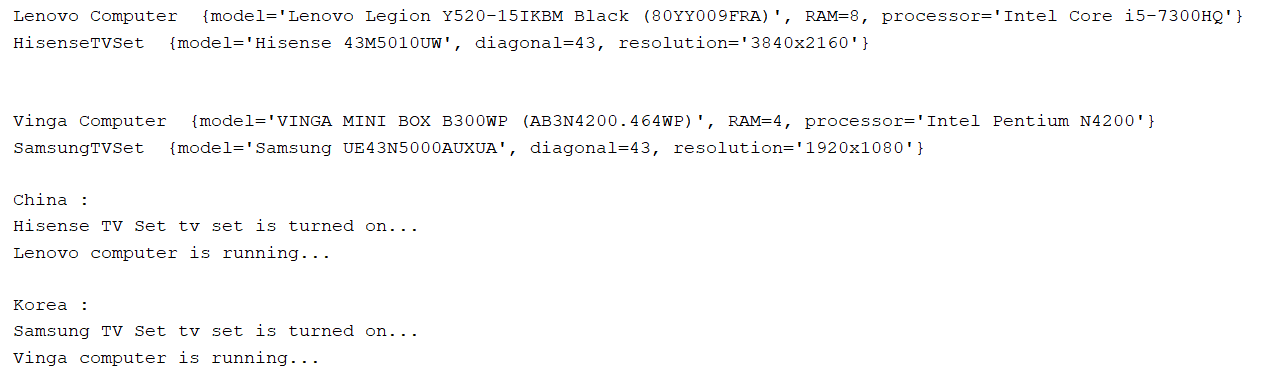
**Результати виконання програми**

Рис. 2. Результат виконання програми

**Висновки**

Отже, за допомогою шаблону “Abstract factory” було створено фабрики різних класів з одним інтерфейсом відповідно до вимог.