Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Современные языки программирования

# ЛАБОРАТОРНАЯ РАБОТА № 1

Вариант № 12

Выполнила:

Ковалева К.В. гр. 310901

Проверил:

Усенко Ф.В.

Минск 2024

**Задание 1:** *Задание к лабораторной работе:*

* согласно варианта задания написать программу (Вариант = номер в списке группы % 30 + 1);
* объекты и их взаимоотношения, имеющиеся в варианте задания, должны быть реализованы;
* функциональная часть приложения, представленная диаграммой последовательности, должна быть реализована;
* разбить функционал приложения на несколько пакетов придерживаясь логики.

Для классов кофе и сахар сделать родительский класс сыпучий предмет. Добавить наследующий от него класс Чай. Добавить класс молоко. Обеспечить класс Чайник методом Нагреть. Классу студент добавить метод Сделать кофе с молоком() и Сделать чай().

import Cup.\*  
  
abstract class AbstractSubject  
{  
 protected var massG: Int = 0;  
}  
class Student : AbstractSubject()  
{  
 public var name : String = "";  
 public fun makeCoffee()  
 {  
 if(*toAddSugar*())  
 {  
 *println*("Вы выбрали сделать кофе с добавлением сахара");  
 val sugar: String = "Сахар";  
 fillCup(sugar, 0);  
 }  
 else  
 {  
 *println*("Вы выбрали сделать кофе без сахара");  
 }  
 val cof = Coffee();  
 *println*("Укажите количество ложек кофе (максимально - 3)");  
 var countofCoffee: Int = 1;  
 try {  
 var choicecoffee : Int = *readln*().*toInt*();  
 if(choicecoffee >=3)  
 {  
 countofCoffee = 3;  
 }  
 else  
 countofCoffee = choicecoffee;  
 }  
 catch (e: Exception)  
 {  
 *println*("Некорректный ввод")  
 countofCoffee = 1;  
 }  
 cof.pour(countofCoffee);  
 val kett = Kettle();  
 kett.fillWater();  
 kett.pourCup(true)  
 }  
 public fun makeTea()  
 {  
 if(*toAddSugar*())  
 {  
 *println*("Вы выбрали сделать чай с добавлением сахара");  
 val sugar: String = "Сахар";  
 fillCup(sugar, 0);  
 }  
 else  
 {  
 *println*("Вы выбрали сделать чай без сахара");  
 }  
 fillCup("Чай", 0)  
 val kett = Kettle();  
 kett.fillWater();  
 kett.pourCup(true)  
 }  
 public fun makeCoffeeMilk()  
 {  
 if(*toAddSugar*())  
 {  
 *println*("Вы выбрали сделать кофе с добавлением сахара");  
 fillCup("Сахар", 0);  
 }  
 else  
 {  
 *println*("Вы выбрали сделать кофе без сахара");  
 }  
 *println*("Укажите количество ложек кофе (максимально - 3)");  
 var countofCoffee: Int = 1;  
 try {  
 var choicecoffee : Int = *readln*().*toInt*();  
 if(choicecoffee >=3)  
 {  
 countofCoffee = 3;  
 }  
 else  
 countofCoffee = choicecoffee;  
 }  
 catch (e: Exception)  
 {  
 *println*("Некорректный ввод")  
 countofCoffee = 1;  
 }  
 fillCup("Кофе", countofCoffee);  
 val kett = Kettle();  
 kett.fillWater();  
 kett.pourCup(false)  
 val milk = Milk()  
 milk.addmilk()  
 }  
 private fun fillCup(something : String, countG: Int)  
 {  
 when(something)  
 {  
 "Сахар"->  
 {  
 val sugar: Sugar = Sugar();  
 *println*("Укажите количество ложек сахара (максимально - 3)");  
 var countofSugar: Int = 1;  
 try {  
 var choicesugar : Int = *readln*().*toInt*();  
 if(choicesugar >=3)  
 {  
 countofSugar = 3;  
 }  
 else  
 countofSugar = choicesugar;  
 }  
 catch (e: Exception)  
 {  
 *println*("Некорректный ввод")  
 countofSugar = 1;  
 }  
 sugar.pour(countofSugar);  
 }  
 "Чай"->  
 {  
 val tea = Tea();  
 tea.pour(countG)  
 }  
 "Кофе"->  
 {  
 val cof = Coffee();  
 cof.pour(countG)  
 }  
 }  
  
 }  
}  
  
  
class Milk: AbstractSubject()  
{  
 public fun addmilk()  
 {  
 *cup*.accept("Молоко", true )  
 }  
}

import Cup.\*  
abstract class GranularSubject : AbstractSubject()  
{  
 abstract public fun pour(countG: Int);  
}  
class Coffee : GranularSubject()  
{  
 private var taste : String = "";  
 override fun pour(countG: Int)  
 {  
 Thread.sleep(1000);  
 *println*("Выберите сорт кофе:")  
 *println*("1. Арабика")  
 *println*("2. Робуста")  
 var time2 : Boolean = true;  
 while(time2) {  
 try {  
 var choice2: Int = *readln*().*toInt*();  
 if (choice2 == 1) {  
 *println*("Вы выбрали сорт: Арабика")  
 taste = "Арабика";  
 time2 = false;  
 }  
 else if (choice2 == 2) {  
 *println*("Вы выбрали сорт: Робуста")  
 taste = "Робуста"  
 time2 = false;  
 }  
 else  
 {  
 *println*("Некорректный ввод");  
 }  
  
 } catch (e: Exception) {  
 *println*("Некорректный ввод");  
 }  
 }  
 *cup*.accept("Кофе", taste, countG)  
 }  
}  
class Tea : GranularSubject()  
{  
 private var sort : String = "";  
 override fun pour(countG: Int)  
 {  
 Thread.sleep(1000);  
 *println*("Выберите сорт кофе:")  
 *println*("1. Черный")  
 *println*("2. Зеленый")  
 var time2 : Boolean = true;  
 while(time2) {  
 try {  
 var choice2: Int = *readln*().*toInt*();  
 if (choice2 == 1) {  
 sort = "Черный";  
 time2 = false;  
 }  
 else if (choice2 == 2) {  
 sort = "Зеленый"  
 time2 = false;  
 }  
 else  
 {  
 *println*("Некорректный ввод");  
 }  
  
 } catch (e: Exception) {  
 *println*("Некорректный ввод");  
 }  
 }  
 *cup*.accept("Чай", sort)  
 }  
}  
class Sugar : GranularSubject()  
{  
 override fun pour(countG: Int)  
 {  
 *cup*.accept("Сахар", countG);  
 }  
}

import Cup.\*  
  
class Kettle: AbstractSubject()  
{  
 private var temperature: Int = 25;  
 private var volumeM : Int = 1000;  
 public fun pourCup(isend : Boolean)  
 {  
 *cup*.accept("Вода",isend );  
 }  
 public fun fillWater()  
 {  
 volumeM = 0;  
 *println*("Вы наполняете чайник водой..");  
 while (volumeM <= 1000)  
 {  
 Thread.sleep(1000)  
 *println*("...")  
 volumeM+=300;  
 }  
 *println*("Вы наполнили чайник водой")  
 Thread.sleep(1000)  
 heat()  
 }  
 public fun heat()  
 {  
 *println*("Чайник нагревается");  
 changeTemperature();  
 *println*("Вода в чайнике закипела.")  
 Thread.sleep(500)  
 }  
 private fun changeTemperature()  
 {  
 while (temperature <= 100)  
 {  
 Thread.sleep(1000)  
 *println*("...")  
 temperature+=30;  
 }  
 }  
}

import AbstractSubject  
  
val *cup* : Cup = Cup();  
class Cup: AbstractSubject()  
{  
 private val volumeMil : Int = 250;  
 public var component : String = "";  
 public fun accept(something: String, count: Int)  
 {  
 if(something == "Сахар")  
 {  
 component += something + " " + count + " ч.л.;";  
 }  
 }  
 public fun accept (something: String, sort: String, countG: Int)  
 {  
 if (something == "Кофе") {  
 component += " " + something + ": Bкус - "+ sort + ", Kол-во ложек - " + countG + ";";  
 }  
 }  
 public fun accept(something: String, sort : String) {  
 if (something == "Чай") {  
 component += " " + something + ": Cорт - "+ sort+ ";";  
 }  
 }  
 public fun accept (something: String, isend : Boolean)  
 {  
 if (something == "Вода") {  
 *println*("Вы налили горячую воду в чашку");  
 Thread.sleep(500)  
 if (isend == true) {  
 *println*("Состав напитка: " + component)  
 component = ""  
 }  
 }  
 if (something == "Молоко") {  
 *println*("Вы налили горячую молоко в чашку");  
 component += " Молоко;"  
 *println*("Состав напитка: " + component)  
 component = ""  
 }  
 }  
}

import Student.\*  
import Kettle.\*  
import Products.\*  
  
fun toAddSugar(): Boolean  
{  
 *println*("Добавить сахар?")  
 *println*("1. Да")  
 *println*("2. Нет")  
 var addsugar : Int = 0  
 addsugar = *readln*().*toInt*();  
 var time : Boolean = true;  
 while(time)  
 {  
 if (addsugar == 1)  
 {  
 time = false;  
 return true;  
 }  
 else if (addsugar == 2)  
 {  
 time = false;  
 return false;  
 }  
 else  
 {  
 *println*("Некорректный ввод")  
 }  
 }  
 return false;  
}  
  
fun main(args: Array<String>){  
 val student = Student()  
 *println*("Введите ваше имя")  
 student.name = *readln*()  
 while (true){  
 *println*()  
 *println*("========================")  
 *println*(student.name + " выберите из списка:")  
 *println*("1. Сделать кофе")  
 *println*("2. Сделать чай")  
 *println*("3. Сделать кофе с молоком")  
 *println*("4. Выход");  
  
 var choice:Number;  
 try{  
 choice = *readln*().*toInt*()  
 }catch (e:Exception){  
 *println*("Некорректный ввод")  
 continue;  
 }  
 when(choice){  
 1->{  
 student.makeCoffee();  
 Thread.sleep(500)  
 *println*("Вы сделали кофе!")  
 Thread.sleep(1500)  
 }  
 2->{  
 student.makeTea()  
 Thread.sleep(500)  
 *println*("Вы сделали чай!")  
 Thread.sleep(1500)  
 }  
 3->{  
 student.makeCoffeeMilk();  
 Thread.sleep(500)  
 *println*("Вы сделали кофе с молоком!")  
 Thread.sleep(1500)  
 }  
 4 -> {  
 break  
 };  
 }  
  
 }  
  
  
}

Результат работы программы представлен на рисунках 1-2.

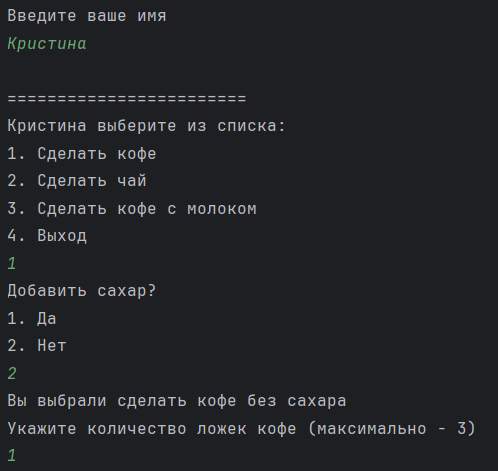


Рисунок 1 – Результат работы программы

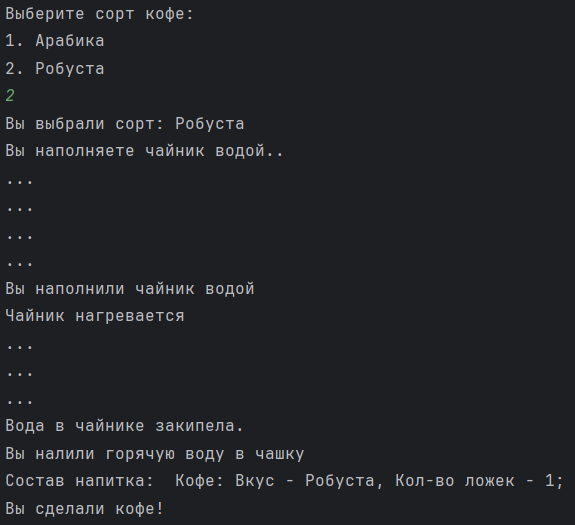


Рисунок 2 – Результат работы программы

Ответы на контрольные вопросы:

* 1. Как работает оператор in для проверки наличия элемента в коллекции?

С его помощью, программа проходит по всем элементам коллекции и проверяет соответствует ли текущий элемент критерию, если да то возвращает элемент.

* 1. Что такое модификаторы доступа (*access modifiers*) в *Kotlin*?

В *Kotlin* предусмотрено четыре модификатора доступа: private, protected, internal и public. Если явно не использовать никакого модификатора, то по умолчанию применяется public. Если вы пометите объявление словом private, оно будет иметь видимость только внутри файла, где было объявлено; Если вы используете internal, видимость будет распространяться на весь [модуль](https://kotlinlang.ru/docs/visibility-modifiers.html#modules); protected запрещено использовать в объявлениях «высокого уровня».

* 1. Сколько конструкторов доступно в *Kotlin*?

В Kotlin доступно несколько типов конструкторов: главный (primary) и один или несколько вторичных (secondary). Главный конструктор объявляется в заголовке класса, а вторичные определяются в теле класса.

**Вывод:**С использованием языка программирования Kotlin создали программу, реализующую взаимодействие разных классов и методов, позволяющих продемонстрировать имитацию работы приготовления кофе, чая, кофе с молоком. Также: сформировали умение разрабатывать программы на языке Kotlin.