Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по программе «**Практическая №10**»

Выполнил: Яковлева Анастасия Николаевна

Группа: ПР-21

Преподаватель: Мирошниченко Г.В

2025

**Вариант 3. Задание:**

1. Создать абстрактный класс и интерфейс в соответствии с вариантом.

2. На их основе описать базовый класс-наследник с указанными в задании полями и методами:

* 2 Конструктора;
* Функция, которая определяет качество объекта – Q по заданной формуле;
* Вывод информации об объекте.

3.Построить класс потомок, который содержит:

* Дополнительное поле Р;
* Функция, которая определяет «качество» объекта класса потомка – Qp, которая перекрывает функцию качества базового класса, выполняя вычисления по новой формуле.

3. Добавить еще 1 класс, который будет использовать функции интерфейса.

4. Создать проект для демонстрации работы: ввод-вывод информации об объектах (не менее 3-х) базового класса и класса потомка.

*Базовый класс:*

Товар на складе:

- наименование;

- цена;

- количество.

Q = цена / количество

*Класс-наследник*

P: год выпуска товара

Qp = Q + 0,5· (Т–Р), где

Т - текущий год

**Входные и выходные данные:**

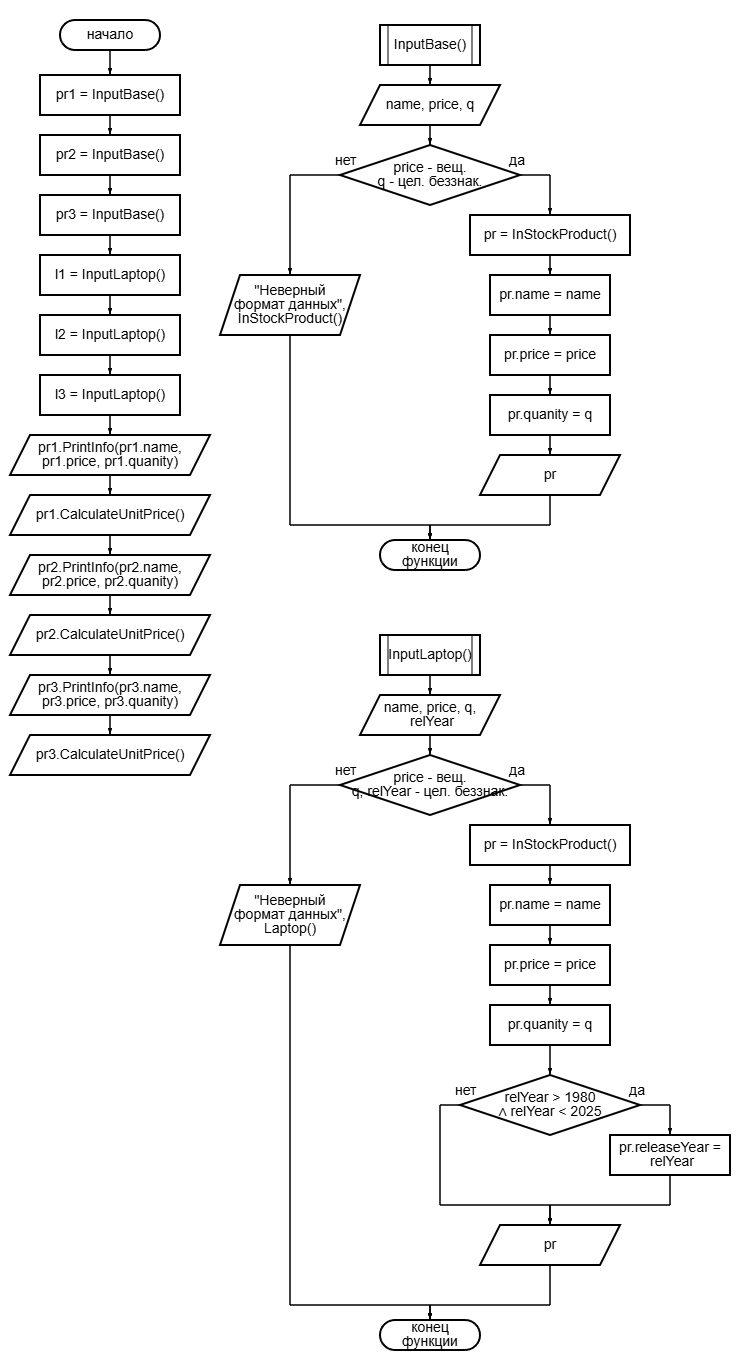
name – наименование товара, строковый;

price – цена, вещественный;

quanity – количество, целочисленный беззнаковый;

relYear – год выпуска товара, целочисленный беззнаковый.

**Блок-схема:**

****

**Листинг программы:**

interface *Info* {  
 fun PrintInfo(name: String, price: Double, quanity: UInt): String  
}

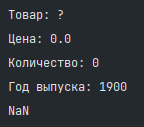
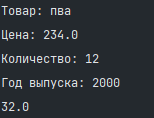
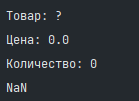
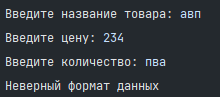
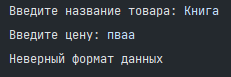
abstract class Product() {  
 abstract val name: String  
 abstract val price: Double  
 abstract val quanity: UInt  
 abstract fun CalculateUnitPrice():Double  
}

open class InStockProduct(): Product(), *Info* {  
 override var name: String = "?"  
 override var price: Double = 0.0  
 override var quanity: UInt = 0u  
 override fun CalculateUnitPrice(): Double {  
 return price/quanity.toDouble()  
 }  
  
 override fun PrintInfo(name: String, price: Double, quanity: UInt): String {  
 return "Товар: $name\nЦена: $price\nКоличество: $quanity"  
 }  
}

class Laptop():InStockProduct() {  
 var releaseYear: UInt = 1900u  
 val currentYear: UInt = 2025u  
 fun a(): Double{  
 return (CalculateUnitPrice() + 0.5\*(currentYear.toDouble()-releaseYear.toDouble()))  
 }  
  
 override fun PrintInfo(name: String, price: Double, quanity: UInt): String {  
 return super.PrintInfo(name, price, quanity) + "\nГод выпуска: $releaseYear"  
 }  
}

fun main() {  
 val pr1 = *InputBase*()  
 val pr2 = *InputBase*()  
 val pr3 = *InputBase*()  
  
 val l1 = *InputLaptop*()  
 val l2 = *InputLaptop*()  
 val l3 = *InputLaptop*()  
  
 *println*(pr1.PrintInfo(pr1.name, pr1.price, pr1.quanity))  
 *println*(pr1.CalculateUnitPrice())  
 *println*(pr2.PrintInfo(pr2.name, pr2.price, pr2.quanity))  
 *println*(pr2.CalculateUnitPrice())  
 *println*(pr3.PrintInfo(pr3.name, pr3.price, pr3.quanity))  
 *println*(pr3.CalculateUnitPrice())  
  
 *println*(l1.PrintInfo(l1.name, l1.price, l1.quanity))  
 *println*(l1.a())  
 *println*(l2.PrintInfo(l2.name, l2.price, l2.quanity))  
 *println*(l2.a())  
 *println*(l3.PrintInfo(l3.name, l3.price, l3.quanity))  
 *println*(l3.a())  
  
  
  
}  
fun InputBase(): InStockProduct  
{  
 try {  
 *print*("Введите название товара: ")  
 val name = *readln*()  
 *print*("Введите цену: ")  
 val price = *readln*().*toDouble*()  
 *print*("Введите количество: ")  
 val q = *readln*().*toUInt*()  
 val pr = InStockProduct()  
 pr.name = name  
 pr.price = price  
 pr.quanity = q  
 *println*()  
 return pr  
 }  
 catch (e: Exception)  
 {  
 *println*("Неверный формат данных")  
 return InStockProduct()  
 }  
}  
fun InputLaptop(): Laptop  
{  
 try {  
 *print*("Введите название товара: ")  
 val name = *readln*()  
 *print*("Введите цену: ")  
 val price = *readln*().*toDouble*()  
 *print*("Введите количество: ")  
 val q = *readln*().*toUInt*()  
 *print*("Введите год выпуска: ")  
 val relYear = *readln*().*toUInt*()  
  
 val pr = Laptop()  
 pr.name = name  
 pr.price = price  
 pr.quanity = q  
 if (relYear > 1980u && relYear < 2025u)  
 pr.releaseYear = relYear  
 *println*()  
 return pr  
 }  
 catch (e: Exception)  
 {  
 *println*("Неверный формат данных")  
 return Laptop()  
 }  
}

**Тестовые ситуации:**

****

**Вывод:** я научилась работать с абстрактными классами и интерфейсами в Kotlin.