МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий
Кафедра технологий программирования

Лабораторная работа №1
по дисциплине: **«Объектно-ориентированные технологии программирования и стандарты проектирования»**
на тему: «**НАСЛЕДОВАНИЕ И ВИРТУАЛЬНЫЕ ФУНКЦИИ**»

ВЫПОЛНИЛ — студент группы 16 ИТ-3
Яблонский А.С.

ПРОВЕРИЛ — преподаватель
Ярошевич П.В.

Полоцк, 2018 г.

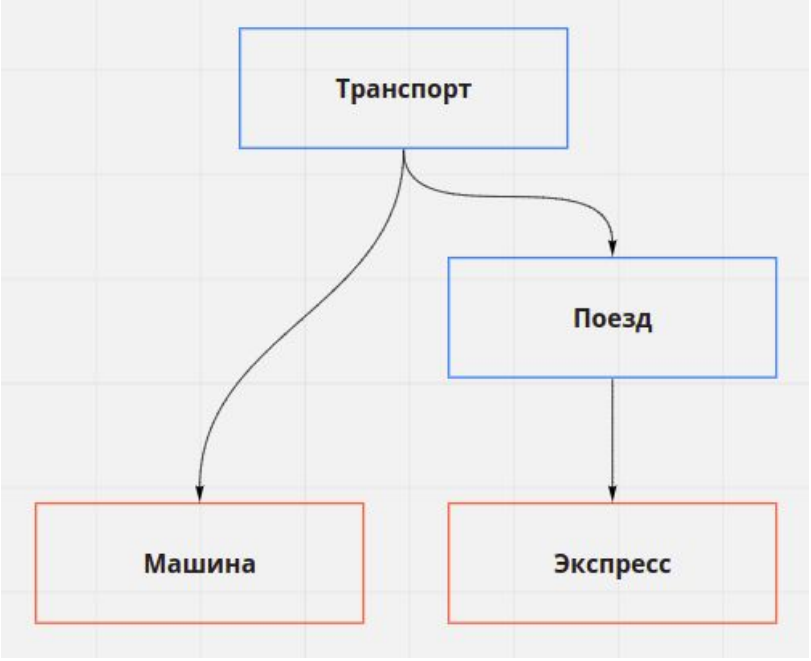
Вариант №11

Постановка задачи:

1. Получить практические навыки создания иерархии классов и использования статических компонентов класса.
2. Написать программу, в которой создается верная иерархия классов - автомобиль, поезд, транспортное средство, экспресс. Включить полиморфные объекты в связанный список, используя статические компоненты класса.
3. Показать использование виртуальных функций. Краткие теоретические сведения.

Реализация:

Для реализации была выбрана следующая иерархическая цепочка:



* в синей рамке - абстрактные классы

1. **Транспорт (*Transport*).**

За основу было принято, что каждое транспортное средство может двигаться с какой-либо скоростью и на определенном виде топливе. Также каждое транспортное средство умеет ехать и сигналить. Исходя из этого было решено реализовать следующие поля и методы в классе Transport:

- поле String fuel - тип топлива
- поле Int speed - скорость движения
- абстрактный метод метод go() - ехать
- метод метод beep() - сигналить, с реализацией

2. **Машина (*Car*)** - прямой наследник класса *Transport.* В классе переопределены set/get методы для полей *speed* и *fuel* и добавлен параметр *String color*. Также

был реализован метод go(). В классе описан единственный конструктор с параметрами *color* и *speed*. Тело класса *Car:*

```
class Car(val color : String, override var speed : Int) : Transport() {

    override var fuel = "PETROL"

    override fun go() {
        println("$color car riding on the road at $speed km/h on $fuel fuel")
    }

}
```

3. **Поезд *(Train)*** - абстрактный класс, прямой наследник класса *Transport.* Переопределен метод beep(). Добавлено поле Int carriageCount - количество вагонов. Тело класса Train:

```
abstract class Train(var carriageCount: Int) : Transport() {

    // final - не может быть переопределен далее
    final override fun beep() {
        println("Tuu-Tuu!")
    }

}
```

4. **Экспресс *(Express)*** - прямой наследник класса Train. Переопределены get/set методы для полей speed и fuel. Реализован метод go(). Реализован конструктор с количеством вагонов в качестве единственного аргумента. Тело класса Express:

```
class Express(carriageCount: Int) : Train(carriageCount) {

    //     constructor(carriageCount: Int) : super(carriageCount)

    override var speed = 75
    override var fuel = "ELECTRICITY"

    override fun go() {
        println("Fast express-train is going on trails with $carriageCount carriage(s) " +
                "at $speed km/h on $fuel fuel")
    }
}
```

5. **Статические компоненты.** Было принято реализовать статические компоненты в классе Transport. В качестве списка, для хранения транспортных средств создано статическая переменная *transportList* типа List<*Transport*>. Для вывода списка транспортных средств на экран реализована статическая

функция printList().

```
@JvmStatic
val transportList = mutableListOf<Transport>()

@JvmStatic
fun printList() {
    for (t in transportList) {
        println("=======")
        t.go()
        t.beep()
    }
}
```

Тестирование:

Тестирование иерархии классов производилось в классе Main, метод main. Для тестирование в статический список транспортных средств класса Transports было добавлено несколько уникальных экземпляров класса Express и Car. После вызвана функция printList(). Код тестирование:

```
Transport.getTransportList().add(new Express(10));
Transport.getTransportList().add(new Car("RED", 60));
Transport.getTransportList().add(new Car("GREEN", 20));
Transport.getTransportList().add(new Express(20));
Transport.getTransportList().add(new Car("BLUE", 85));

Transport.printList();
```

В результате работы данного кода получили следующий результат:

=======
Fast express-train is going on trails with 10 carriage(s) at 75 km/h on ELECTRICITY fuel
Tuu-Tuu!
=======
RED car riding on the road at 60 km/h on PETROL fuel
Beep-Beep!
=======
GREEN car riding on the road at 20 km/h on PETROL fuel
Beep-Beep!
=======
Fast express-train is going on trails with 20 carriage(s) at 75 km/h on ELECTRICITY fuel
Tuu-Tuu!
=======
BLUE car riding on the road at 85 km/h on PETROL fuel
Beep-Beep!

Результаты тестирования соответствовали ожидаемым. Тестирование прошло успешно.

Ввод вывод полей. Для ввода полей класса использовался объект класса Scanner и его методы nextLine(), nextInt(), nextLong().

Пример метода ввода поля ID:

```
fun readId() {
    val scanner = Scanner(System.`in`)
    print("Enter company ID: ")
    this.id = scanner.nextInt()
}
```

Для вывода полей использовался методы print() и println(). Пример вывода поля address:

```
fun printAddress() {
    println("Company address: ${this.address}")
}
```

6. **Конструкторы.**
   1) В конструктор класса с параметрами, в качестве параметров передавались значения ID и название фирмы. В результате конструктор принял следующий вид:

      ```
      constructor(id: Int, name: String) {
          this.id = id
          this.name = name
      }
      ```

      В теле конструктора, переданные ему значения присваиваются соответствующим полям объекта.

   2) Конструктор копирования был реализован следующим образом

      ```
      constructor(company: Company) {
          this.id = company.id
          this.name = company.name
          this.address = company.address
          this.tel = company.tel
          this.owner = company.owner
          this.createDate = company.createDate
      }
      ```

      Данный конструктор принимает единственный аргумент - объект класса Company. После устанавливает все поля текущего объекта (this) соответственным поля объекта, переданного в аргументах

   3) Конструктор без параметров:

      ```
      constructor()
      ```

Вывод:

В результате выполнения данной лабораторной работы мне удалось получить практические навыки создания иерархии классов и использования статических компонентов класса. Мне удалось написать программу, в которой создается иерархия

классов. Включить полиморфные объекты в список, используя статические компоненты класса. Показать использование абстрактных функций.