

Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Объектно-ориентированное программирование

ОТЧЁТ

к лабораторной работе
на тему

Создание объектной модели предметной сущности

Выполнил: студент группы 053506
Слуцкий Никита Сергеевич

Проверил: Летохо Александр Сергеевич

Минск 2022

Оглавление

Цели выполнения задания	3
Краткие теоретические сведения.....	3
Задание	4
Результат выполнения	4
Вставки кода	6

Цели выполнения задания:

Получить опыт практической работы в создании иерархий классов предметной области с помощью UML диаграмм и изучить паттерн фабрики.

Краткие теоретические сведения

Фабричный метод — это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов

Паттерн “Фабричный метод” предлагает создавать объекты не напрямую, используя оператор new, а через вызов особого фабричного метода. Объекты всё равно будут создаваться при помощи new, но делать это будет этот фабричный метод.

Чтобы эта система заработала, все возвращаемые объекты должны иметь общий интерфейс (наследоваться от одного абстрактного класса). Подклассы смогут производить объекты различных классов, следующих одному и тому же интерфейсу.

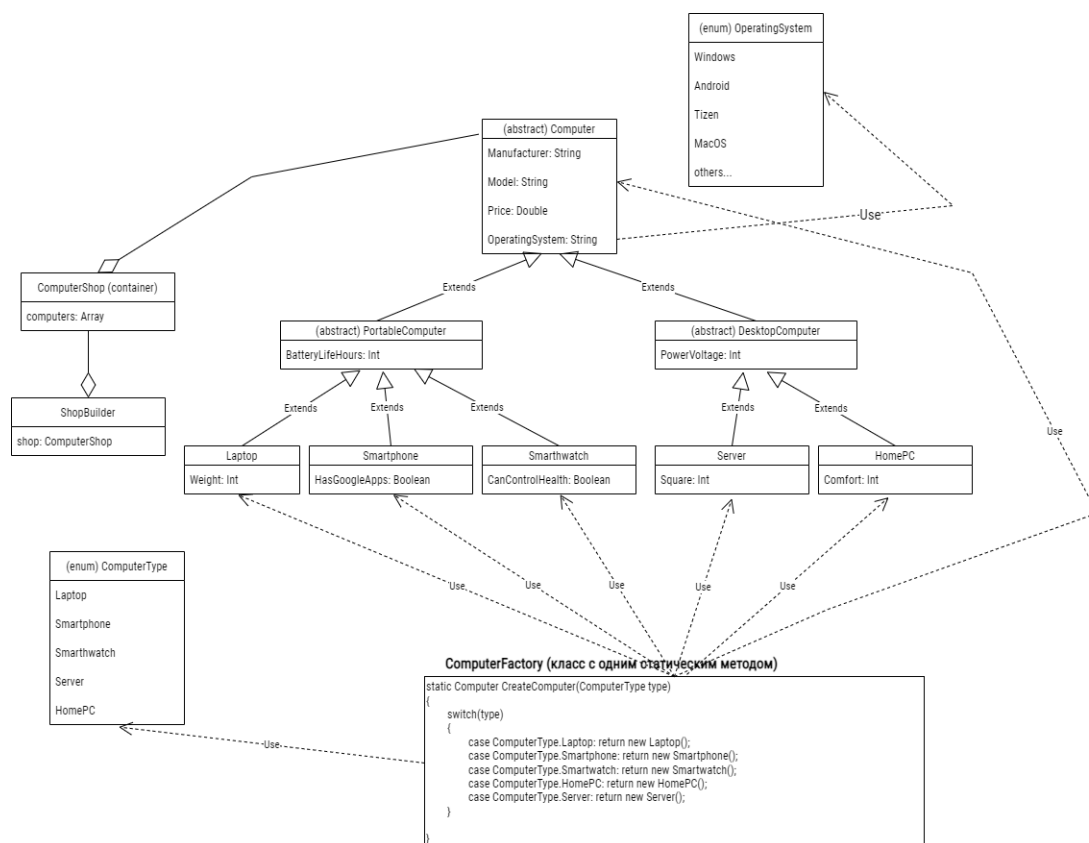
Например, классы “Грузовик” и “Судно” реализуют интерфейс “Транспорт” с методом “доставить” (или наследуются от одного и того же абстрактного класса “Транспорт”). Каждый из этих классов реализует метод по-своему: грузовики везут грузы по земле, а суда — по морю. Или же каждый имеет свои дополнительные поля. Фабричный метод по переданному значению перечисления будет возвращать разные объекты. Их можно поместить, например, в один массив с типом “Транспорт”.

Задание

Необходимо описать иерархию классов предметной области в виде UML диаграммы и классов на языке Java, Kotlin или другом, предложенном преподавателем, в соответствии с выбранным вариантом задания. Предметная сущность – “Компьютеры”.

Ход выполнения:

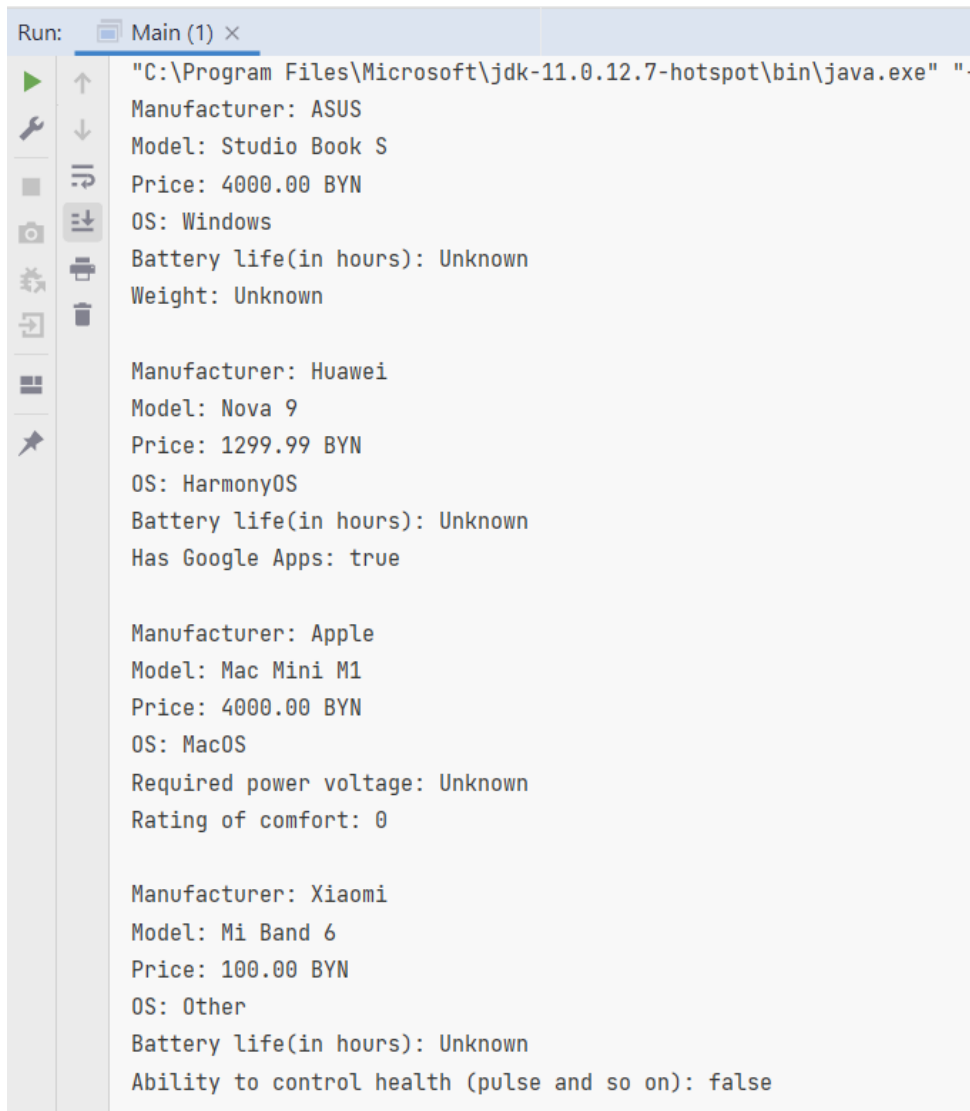
UML-диаграмма, составленная для предметной области “Компьютеры”:



Результат вывода программы

Разработано в среде IntelliJ IDEA на языке программирования Java. В ходе программы фабрикой создаются “пустые” экземпляры

различных классов из низа иерархии и формируют экземпляр класса “Магазин компьютеров”, где хранятся в списке с типом Computer, то есть абстрактным типом. Программа добавляет в магазин начальные значения с помощью фабричного метода класса ComputerFactory, а также заполняет некоторыми данными эти экземпляры. Далее происходит вывод всего, что было добавлено в магазин.



```
Run: Main (1) x
"C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe" "-
Manufacturer: ASUS
Model: Studio Book S
Price: 4000.00 BYN
OS: Windows
Battery life(in hours): Unknown
Weight: Unknown

Manufacturer: Huawei
Model: Nova 9
Price: 1299.99 BYN
OS: HarmonyOS
Battery life(in hours): Unknown
Has Google Apps: true

Manufacturer: Apple
Model: Mac Mini M1
Price: 4000.00 BYN
OS: MacOS
Required power voltage: Unknown
Rating of comfort: 0

Manufacturer: Xiaomi
Model: Mi Band 6
Price: 100.00 BYN
OS: Other
Battery life(in hours): Unknown
Ability to control health (pulse and so on): false
```

Вставки кода классов из иерархии, а также кода класса для фабричного метода

```
public abstract class Computer {
    protected static final String DefaultCurrency = "BYN";
    protected static final String DefaultValue = "Unknown";

    private String manufacturer;
    private String model;
    private double price;
    private OperatingSystem operatingSystem;

    protected Computer() {
        this.manufacturer = Computer.DefaultValue;
        this.model = Computer.DefaultValue;
        this.price = 0.00;
        this.operatingSystem = OperatingSystem.NoOS;
    }

    protected Computer(String manufacturer, String model, double price, OperatingSystem os) {
        this.manufacturer = manufacturer;
        this.model = model;
        this.price = price;
        this.operatingSystem = os;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setOperatingSystem(OperatingSystem operatingSystem) {
        this.operatingSystem = operatingSystem;
    }

    public String toString() {
        return String.format(
            "Manufacturer: %s\nModel: %s\nPrice: %.2f %s\nOS: %s",
            this.manufacturer,
            this.model,
            this.price,
            Computer.DefaultCurrency,
            this.operatingSystem.name()
        );
    }
}



---



public abstract class PortableComputer extends Computer {
    private byte batteryLifeHours;

    protected PortableComputer() {
        super();
        this.batteryLifeHours = 0;
    }

    protected PortableComputer(String manufacturer, String model, double price, OperatingSystem os, byte
batteryLife) {
        super(manufacturer, model, price, os);
        this.batteryLifeHours = batteryLife;
    }
}
```

```

    public String toString() {
        return String.format(
            "%s\nBattery life(in hours): %s",
            super.toString(),
            this.batteryLifeHours == 0.00 ? Computer.DefaultValue : String.valueOf(this.batteryLifeHours)
        );
    }
}

```

```

public abstract class DesktopComputer extends Computer {
    private int powerVoltage;

    protected DesktopComputer() {
        super();
        this.powerVoltage = 0;
    }

    protected DesktopComputer(String manufacturer, String model, double price, OperatingSystem os, int
powerVoltage) {
        super(manufacturer, model, price, os);
        this.powerVoltage = powerVoltage;
    }

    public String toString() {
        return String.format(
            "%s\nRequired power voltage: %s",
            super.toString(),
            this.powerVoltage == 0.00 ? Computer.DefaultValue : String.valueOf(this.powerVoltage)
        );
    }
}

```

```

public class Laptop extends PortableComputer {
    private double weight;

    public Laptop(String manufacturer, String model, double price, OperatingSystem os, byte batteryLife, double
weight) {
        super(manufacturer, model, price, os, batteryLife);
        this.weight = weight;
    }

    public Laptop() {
        super();
        this.weight = 0.00;
    }

    public String toString() {
        return String.format(
            "%s\nWeight: %s",
            super.toString(),
            this.weight == 0.0 ? Computer.DefaultValue : String.valueOf(this.weight)
        );
    }
}

```

```

public class Smartphone extends PortableComputer {
    private boolean hasGoogleApps;

    public Smartphone() {
        super();
        this.hasGoogleApps = true;
    }

    public Smartphone(String manufacturer, String model, double price, OperatingSystem os, byte batteryLife,
boolean hasGapps) {
        super(manufacturer, model, price, os, batteryLife);
        this.hasGoogleApps = hasGapps;
    }

    public String toString() {
        return String.format(

```

```

        "%s\nHas Google Apps: %b",
        super.toString(),
        this.hasGoogleApps
    );
}
}

```

```

public class Smartwatch extends PortableComputer {
    private boolean canControlHealth;

    public Smartwatch(String manufacturer, String model, double price, OperatingSystem os, byte batteryLife,
boolean canControlHealth) {
        super(manufacturer, model, price, os, batteryLife);
        this.canControlHealth = canControlHealth;
    }

    public Smartwatch() {
        super();
        this.canControlHealth = false;
    }

    public String toString() {
        return String.format(
            "%s\nAbility to control health (pulse and so on): %b",
            super.toString(),
            this.canControlHealth
        );
    }
}

```

```

public class Server extends DesktopComputer{
    private int square;

    public Server() {
        super();
        this.square = 0;
    }

    public Server(String manufacturer, String model, double price, OperatingSystem os, int powerVoltage, int
square) {
        super(manufacturer, model, price, os, powerVoltage);
        this.square = square;
    }

    public String toString() {
        return String.format("%s\nRequired free space (square): %d", super.toString(), this.square);
    }
}

```

```

public class HomePC extends DesktopComputer {
    private int comfortRating;

    public HomePC() {
        super();
        this.comfortRating = 0;
    }

    public HomePC(String manufacturer, String model, double price, OperatingSystem os, int powerVoltage, int
comfortRating) {
        super(manufacturer, model, price, os, powerVoltage);
        this.comfortRating = comfortRating;
    }

    public String toString() {
        return String.format("%s\nRating of comfort: %d", super.toString(), this.comfortRating);
    }
}

```

```
public class ComputerFactory {
    public static Computer CreateComputer(ComputerType type) {
        switch (type) {
            case HomePC:
                return new HomePC();
            case Server:
                return new Server();
            case Smartwatch:
                return new Smartwatch();
            case Smartphone:
                return new Smartphone();
            case Laptop:
                return new Laptop();
            default:
                throw new RuntimeException("Unhandled error. Unknown computer's type");
        }
    }
}
```