**Домашняя работа Incapsulation**

**Задание 1.** Напишите код класса с применением принципа инкапсуляции

Создайте проект и создайте в нём класс Main с методом main.

В рамках этого задания вам нужно создать класс ChildBankAccount, который будет контролировать работу детского карточного банковского счёта:

Создайте в этом классе две переменные: одну balance, в которой будет храниться баланс, и вторую maxBalance, в которой будет храниться максимально возможный баланс:

Создайте конструктор с параметром maxBalance и устанавливайте значение переменной maxBalance из этого параметра. Это будет максимальный баланс, на который можно пополнить счёт.

Создайте в классе метод пополнения карточного счёта с параметром value, содержащим количество денег, на которое необходимо пополнить счёт:

public boolean depositMoney(double value) {

}

Метод должен защищать balance от ошибочных изменений — уменьшения (в случае, если value меньше нуля) и увеличения сверх лимита (выше значения maxBalance).

В методе напишите проверку того, что сумма счёта после пополнения не станет выше максимально допустимой — выше значения переменной maxBalance. Если это условие выполняется, увеличьте баланс счёта на значение value и верните true. Если условие не выполняется или переменная value меньше нуля, метод должен возвращать false.

Создайте метод списания денег со счёта:

public boolean debitMoney(double value) {

}

Этот метод должен защищать переменную balance от увеличения (в случае, если будет передано значение value меньше 0) и от уменьшения ниже нуля.

Напишите в этом методе проверку того, что баланс счёта не станет отрицательным, если из него вычесть value, а также проверку того, что value — неотрицательное число. В случае, если всё в порядке, метод должен вернуть true. Если хотя бы одно из условий не выполняется, метод должен вернуть false.

Создайте также в классе ChildBankAccount метод getBalance, который будет возвращать текущий баланс счёта.

Напишите в методе main класса Main код, который будет проверять корректность реализации методов класса ChildBankAccount:

ChildBankAccount account =

new ChildBankAccount(10000);

account.depositMoney(1000);

account.depositMoney(2000);

account.depositMoney(10000);

account.depositMoney(-1000);

System.out.println("Balance: " + account.getBalance());

account.debitMoney(500);

account.debitMoney(422.75);

account.debitMoney(50000);

account.debitMoney(-50);

System.out.println("Balance: " + account.getBalance());

Выполните получившийся код и сверьте результат с эталонным:

Balance: 3000.0

Balance: 2077.25

Если что-то не получилось, постарайтесь самостоятельно внести исправления в свой код и добиться необходимого результата — вывода в консоль текста, полностью идентичного показанному выше.

Если всё получилось, поздравляем! Вы написали код класса, в котором значение переменной balance защищено от некорректных изменений, а работа с этой переменной осуществляется скрытым в методах depositMoney и debitMoney кодом. Это наглядная демонстрация принципа инкапсуляции, которого вам будет необходимо придерживаться в будущем, создавая свои классы и методы.

**Задание 2\*.** Создайте класс для хранения данных о странах (POJO)

Создайте класс для хранения данных о странах

Создайте класс Country, который будет использоваться для хранения данных о странах, со следующими полями:

* название (класс String);
* численность населения (тип int);
* площадь в квадратных километрах (тип double);
* название столицы (класс String);
* наличие выхода к морю (тип boolean).

Имена полей установите самостоятельно так, чтобы они были понятными и соответствовали хранящимся в них данным.

Создайте также в этом классе:

* конструктор с одним параметром — названием страны;
* методы-геттеры для всех полей класса;
* методы-сеттеры для всех полей класса.

В методе main класса Main напишите код, проверяющий работу хотя бы двух геттеров и сеттеров в этом классе. Запустите этот код и убедитесь, что он работает верно.

Если что-то не получилось, постарайтесь самостоятельно внести исправления в свой код так, чтобы он стал работать.

Если всё получилось, поздравляем! Теперь вы умеете создавать POJO-классы, геттеры и сеттеры!

**Задание 3\*.** Модифицируйте класс и добавьте в него неизменяемое поле

Создайте новый проект Immutability и создайте в нём класс Product. Напишите в нём такой же код:

public class Product {

private final String name;

private final int price;

public Product(String name, int price) {

this.name = name;

this.price = price;

}

public String getName() {

return name;

}

public int getPrice() {

return price;

}

}

Сделайте поле price изменяемым: уберите ключевое слово final перед объявлением переменной и добавьте в класс сеттер для этого поля. Также уберите его из конструктора, класс при этом перестанет быть иммутабельным.

Добавьте в этот класс ещё одно неизменяемое поле String barCode и метод-геттер для него. В нём будет храниться штрихкод товара. Добавьте его также в конструктор.

Создайте класс Main с методом main и напишите в нём код, который будет использовать класс Product — создавать объект этого класса, менять в нём значение переменной price и выводить новое значение в консоль.

Если что-то не получилось, постарайтесь самостоятельно внести исправления в свой код так, чтобы он стал работать.

Если всё получилось, поздравляю! Теперь вы умеете управлять изменяемостью полей классов!