Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина Программирование мобильных информационных систем

Отчет

по лабораторной работе

на тему:

**«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП)»**

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Булатевич В.Ю.

(подпись)

гр. 214301

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Усенко Ф.В.

(подпись)

Минск, 2024

**Цель работы:** Изучить принципы объектно-ориентированного программирования (ООП) в Kotlin, включая классы, объекты, наследование, полиморфизм, интерфейсы и абстрактные классы. Научиться создавать и использовать собственные классы, а также применять принципы ООП на практике.

1. **Задание**: Разработайте сложную банковскую систему, включающую классы BankAccount, CreditAccount, SavingsAccount, и InvestmentAccount. Реализуйте методы для выполнения транзакций, учета процентов и капитализации, а также методы для анализа и прогнозирования финансового состояния счетов клиентов.

**1 Реализация задания**

open class BankAccount(val accountNumber: String, var balance: Double) {  
  
 open fun transfer(amount: Double, toAccount: BankAccount) {  
 if (balance >= amount) {  
 balance -= amount  
 toAccount.balance += amount  
 } else {  
 throw IllegalArgumentException("Недостаточно средств на счете")  
 }  
 }  
ф  
 open fun deposit(amount: Double) {  
 if (amount > 0) {  
 balance += amount  
 *println*("Счет пополнен на $amount. Новый баланс: $balance")  
 } else {  
 *println*("Сумма пополнения должна быть положительной.")  
 }  
 }  
  
 open fun withdraw(amount: Double) {  
 if (amount <= 0) throw IllegalArgumentException("Сумма снятия должна быть больше нуля")  
 if (amount > balance) throw IllegalArgumentException("Недостаточно средств")  
 balance -= amount  
 *println*("Снятие $amount. Новый баланс: $balance")  
 }  
  
 open fun displayBalance() {  
 *println*("Баланс счета: $balance")  
 }  
}

**2 Ответы на контрользые вопросы**

1. Что такое класс в Kotlin, и как он объявляется?  
   Класс в Kotlin — это основная единица, объединяющая данные и функции для их обработки. Он объявляется с использованием ключевого слова class, например, class Person.
2. Как создать объект класса в Kotlin? Приведите пример.  
   Объект создаётся с использованием имени класса и круглых скобок. Например, val person = Person().
3. Что такое свойства класса, и как их объявить в Kotlin?  
   Свойства класса — это переменные, описывающие состояние объекта. Они объявляются внутри класса с использованием ключевых слов val или var. Например, var name: String = "John".
4. Как объявить и использовать метод класса? Приведите пример.  
   Метод класса — это функция, объявленная внутри класса. Например, в классе Person можно объявить метод greet: fun greet() = "Hello!", а затем вызвать его через объект: person.greet().
5. Что такое первичный конструктор, и как он используется для инициализации свойств класса?  
   Первичный конструктор — это часть заголовка класса, где можно объявить инициализацию свойств. Например, в классе Person: class Person(val name: String).
6. Как в Kotlin создать вторичный конструктор, и зачем он может понадобиться?  
   Вторичный конструктор создаётся с использованием ключевого слова constructor. Он нужен, чтобы создать объект с разными наборами параметров. Например, constructor(name: String, age: Int).
7. Что такое наследование, и как его реализовать в Kotlin? Приведите пример.  
   Наследование позволяет одному классу унаследовать свойства и методы другого. Например, класс Dog может наследовать класс Animal с помощью записи class Dog : Animal().
8. Как переопределить метод базового класса в подклассе? Приведите пример.  
   Метод переопределяется с использованием ключевого слова override. Например, переопределить метод sound базового класса Animal в подклассе Dog: override fun sound() = "Bark".
9. В чем разница между интерфейсом и абстрактным классом в Kotlin?  
   Интерфейсы не могут содержать состояния (свойств с реализацией), но могут иметь абстрактные и реализованные методы. Абстрактные классы могут иметь состояние и используются для создания базовых классов с частично реализованным функционалом.
10. Как реализовать множественное наследование через интерфейсы? Приведите пример.  
    Для реализации множественного наследования класс может реализовать несколько интерфейсов, например, class C : A, B. В случае конфликтов методов можно явно указать, какой метод вызывать, например, super<A>.method().

**Вывод**

В процессе изучения классов и объектов в Kotlin были рассмотрены основные механизмы работы с ними. Было освоено создание классов, объектов, методов и свойств, что позволяет моделировать реальный мир в программировании. Изучение первичных и вторичных конструкторов дало понимание гибких способов инициализации объектов.