МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Факультет
Информатика и вычислительная техника

Кафедра
Программное обеспечение вычислительной техники и автоматизированных систем

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ: КОЛЛЕКЦИИ И ЛЯМБДЫ

Практикум

по выполнению лабораторной работы №3
по дисциплине «Языки программирования высокого уровня»

Технология разработки сложных программных систем
(указывается направленность (профиль) образовательной программы)

09.04.04 Программная инженерия
(указывается код и наименование направления подготовки)

Ростов-на-Дону
2024 г.

Составители: канд. техн. наук, зав. каф. В.В. Долгов

УДК 004.432

Базовые типы данных, управляющие конструкции и функции: метод. указания. – Ростов н/Д: Издательский центр ДГТУ, 2024.

В методическом указании рассматриваются вопросы создания data-классов, работа со стандартными коллекциями языка Kotlin и использование анонимных функций и функций высшего порядка для обработка данных. Приведены задания к лабораторной работе, помогающие закрепить на практике полученные знания, и контрольные вопросы для самопроверки.
Предназначено для обучающихся по направлению 09.04.04 «Программная инженерия».

Ответственный за выпуск:
зав. кафедрой «Программное обеспечение вычислительной техники и автоматизированных систем» Долгов В.В.

1 Теоретическая часть

Kotlin – современный статически типизированный язык программирования, разработанный компанией JetBrains. Он компилируется в байт-код JVM, что позволяет использовать его для разработки под платформу Java, но при этом он обладает более лаконичным синтаксисом и современными возможностями. В данной лабораторной работе мы рассмотрим особенности data-классов, работу со стандартными коллекциями Kotlin и использование анонимных функций и функций высшего порядка для обработки данных.

1.1 Data-классы в Kotlin

Data-классы предназначены для хранения данных. Они позволяют автоматически генерировать полезные методы, такие как equals(), hashCode(), toString(), а также функции для работы с компонентами и метод copy().

Чтобы объявить data-класс, достаточно добавить ключевое слово data перед объявлением класса:

```
data class Person(val name: String, val age: Int)
```

В приведенном выше примере Person – data-класс с двумя неизменяемыми свойствами: name и age.

Автоматически сгенерированный метод *toString* позволяет получать строковое представление объекта, выводя значения его свойств:

```
val person = Person("Alice", 30)
println(person)            // Вывод: Person(name=Alice, age=30)
```

А методы equals() и hashCode() позволяют сравнивать объекты по значению их свойств, а не по ссылкам:

```
val person1 = Person("Bob", 25)
val person2 = Person("Bob", 25)
println(person1 == person2) // Вывод: true
```

В то же время стоит обратить внимание, что все автоматически сгенерированные методы оперируют и принимают во внимание только те свойства, которые объявлены в заголовке класса как параметры главного конструктора. Свойства, объявленные в теле такого класса, не будут попадать в строку, генерируемую методом *toString* и не будут приниматься во внимание при сравнении объектов, если только программист не реализует такой функционал самостоятельно.

Помимо этого, data-классы могут быть разбиты на компоненты через деструктуризацию:

```
val (name, age) = person
println("$name is $age years old") // Вывод: Alice is 30 years old
```

и скопированы с использованием методы *copy*, который создает копию объекта с возможностью изменения некоторых свойств:

```
val newPerson = person.copy(age = 31)
println(newPerson) // Вывод: Person(name=Alice, age=31)
```

Data-классы существенно сокращают количество шаблонного кода, облегчая работу с объектами, которые служат для передачи данных. Это делает код более читаемым и поддерживаемым. Таким образом, data-классы идеально подходят для хранения данных, передачи их между функциями и модулями, а также для сериализации и десериализации.

Предоставляя ряд удобных особенностей, data-классы в то же время имеют ряд ограничений и требований, накладываемых на них, а именно:

- Первичный конструктор data-класса должен иметь хотя бы один параметр.
- Все параметры первичного конструктора должны быть помечены как val или var.
- Data-классы не могут быть abstract, open, sealed или inner.

1.2 Стандартные коллекции Kotlin

Kotlin предоставляет богатый набор коллекций, разделенных на изменяемые и неизменяемые, что позволяет разработчику выбрать наиболее подходящий вариант для конкретной задачи.

Неизменяемые (immutable) коллекции не позволяют изменять свое содержимое после их создания. Представлены такими интерфейсами как *List*, *Set* и *Map*.

Изменяемые (mutable) коллекции позволяют изменять свое содержимое после создания. Представлены интерфейсами *MutableList*, *MutableSet*, *MutableMap*.

Примеры создания неизменяемых и изменяемых коллекций приведены в примере ниже.

```
// Неизменяемые коллекции
val numbers = listOf(1, 2, 3)
val names = setOf("Alice", "Bob")
val userAges = mapOf("Alice" to 30, "Bob" to 25)
```

```
// Изменяемые коллекции
val mutableNumbers = mutableListOf(1, 2, 3)
val mutableNames = mutableSetOf("Alice", "Bob")
val mutableUserAges = mutableMapOf("Alice" to 30, "Bob" to 25)
```

В то же время, язык программирования Kotlin предоставляет богатый набор функций высшего порядка для работы с коллекциями, таких как фильтрация, преобразования элементов и различного рода аггрегации.

```
// выбор в коллекции только четных чисел
val evenNumbers = numbers.filter { it % 2 == 0 }

// расчет квадратов для каждого числа в коллекции
val squaredNumbers = numbers.map { it * it }

// пример расчета суммы чисел (на самом деле лучше использовать метод sum)
val sum = numbers.reduce { acc, number -> acc + number }
```

1.3 Анонимные функции и функции высшего порядка

Анонимные функции, или лямбда-выражения, позволяют определять функции без имен, которые можно передавать как параметры или возвращать из других функций.

```
val sum = { x: Int, y: Int -> x + y }
println(sum(2, 3)) // Вывод: 5
```

Особенно полезными лямбда-выражения становятся совместно с функциями высшего порядка, которые принимают другую функцию в качестве параметра или же возвращают функцию.

```
fun performOperation(a: Int, b: Int, operation: (Int, Int) -> Int): Int
{
    return operation(a, b)
}

val result = performOperation(5, 3) { x, y -> x * y }
println(result)        // Вывод: 15
```

Предположим, у нас есть список чисел, и требуется выполнить над ними ряд последовательных операций: отфильтровать четные числа,

возвести их в квадрат, отсортировать в порядке убывания и вывести результат.

Тогда, применяя функции обработки коллекций и лямбда-выражения – это можно записать в виде единого последовательного вызова соответствующих функций, что делает код нагляднее и понятнее.

```
fun main() {
    val numbers = listOf(5, 2, 7, 8, 1, 4, 6)

    val processedNumbers = numbers
        .filter { it % 2 == 0 }
        .map { it * it }
        .sortedDescending()

    println(processedNumbers) // Вывод: [64, 36, 16, 4]
}
```

2 Задание к лабораторной работе

Создать в одной из сред программирования (на выбор предлагается IntelliJ IDEA Community Edition или Visual Studio Code [3]) проект консольного приложения на языке Kotlin. Используя функции ввода/вывода информации, реализовать программу в соответствии с вариантом задания (табл. 2). В процессе выполнения работы запрещено использовать стандартные функции или библиотеки языка программирования Kotlin либо среды выполнения JVM, выполняющие значимую часть функционала задания.

Код программы должен содержать обработку ошибочных ситуаций, которые могут возникнуть в ходе выполнения программы. В случае возникновения ошибок, организовать информативный вывод данных о возникшей ошибке на экран с предложением продолжить выполнение, проигнорировав ошибку, или завершить выполнение программы.

При защите работы студент должен уметь: создавать консольные проекты, устанавливать точки останова для отладки программы, выполнять программу пошагово в режиме отладки, просматривать значения переменных при отладке.

Таблица 2. Варианты заданий

№ вари-анта	Задание к лабораторной работе
1	Разработка системы управления библиотекой. 1) Создайте связанные между собой data-классы для представления книг, авторов и читателей: - Книга: содержит информацию о названии, авторе, годе издания, жанре. - Автор: включает имя, дату рождения, список книг. - Читатель: содержит имя, номер читательского билета, список взятых книг. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Добавления новых книг и авторов в каталог. - Регистрации новых читателей. - Выдачи и возврата книг читателями с проверкой наличия книги в библиотеке и у читателя. - Поиска книг по названию, автору, жанру. - Отслеживания текущей доступности книг. - Подсчёта количества книг у каждого читателя. 3) Создайте приложение, позволяющее взаимодействовать с системой: добавлять книги, регистрировать читателей, выдавать и возвращать книги, искать книги по различным критериям.
2	Моделирование системы бронирования авиабилетов. 1) Разработайте следующие data-классы: - Рейс: номер рейса, пункт отправления и прибытия, дата и время, список мест. - Пассажир: имя, паспортные данные, список билетов. - Билет: привязан к рейсу и месту. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Поиска рейсов по заданным параметрам (дата, маршрут). - Бронирования и отмены билетов пассажирами. - Проверки доступности мест на рейсах. - Фильтрации рейсов по дате, маршруту, наличию мест. - Управления списком забронированных мест. - Подсчёта количества доступных мест на рейсе. 3) Создайте приложение, позволяющее пассажирам искать рейсы по различным параметрам и бронировать билеты.
3	Система управления заказами в интернет-магазине.

	1) Создайте связанные между собой data-классы: - Продукт: название, категория, цена, количество на складе. - Клиент: имя, контактные данные, история заказов. - Заказ: список продуктов, клиент, статус заказа. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Добавления продуктов в каталог, обновления информации о наличии. - Регистрации клиентов. - Оформления заказов клиентами с обновлением остатков на складе. - Расчёта общей стоимости заказа с учётом возможных скидок. - Поиска продуктов по категории, цене. - Управления корзиной покупателя. - Анализа продаж по категориям продуктов. 3) Создайте приложение, эмулирующее процесс покупки в интернет-магазине, включая просмотр каталога, добавление товаров в корзину и оформление заказа.
4	Система управления работой ресторана. 1) Разработайте следующие data-классы: - Меню: список блюд с ценами и временем приготовления. - Столик: номер, количество мест, статус (свободен, занят). - Заказ: столик, список блюд, время заказа, статус выполнения. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Бронирования столиков. - Размещения и изменения заказов. - Расчёта счёта для клиента. - Поиска популярных блюд (на основе частоты заказов). - Отслеживания времени приготовления заказов. - Управления очередью заказов на кухне. 3) Создайте приложение, эмулирующее работу ресторана, с возможностью оформления заказов и управления столиками.
5	Система управления проектами. 1) Создайте следующие data-классы: - Проект: название, описание, список задач, команда. - Задача: описание, исполнитель, статус, приоритет, сроки. - Сотрудник: имя, роль, список задач. 2) Используя коллекции и лямбда-выражения, создайте функции для:

	- Создания проектов и задач. - Назначения задач сотрудникам. - Обновления статусов задач. - Сортировки задач по приоритету, срокам. - Фильтрации задач по статусу, исполнителю. - Анализа нагрузки на сотрудников. 3) Создайте приложение, помогающее управлять проектами, распределять задачи и отслеживать их выполнение.
6	Анализ данных о погоде. 1) Разработайте следующие data-классы: - Запись о погоде: дата, город, температура, влажность, осадки. - Город: название, страна, координаты, список записей погоды. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Сбора и хранения данных о погоде. - Анализа погодных тенденций. - Вычисления средних значений температуры и осадков за период. - Поиска экстремальных значений. - Сравнения погодных условий между городами. 3) Создайте приложение, представляющее наполнять базу данными о погоде, создавать и выводить отчёты по погодным данным, позволяя пользователям изучать исторические данные и тенденции.
7	Система управления библиотекой музыки. 1) Создайте следующие data-классы: - Артист: имя, жанр, список альбомов. - Альбом: название, год выпуска, список треков. - Трек: название, длительность, рейтинг. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Создания и управления плейлистами. - Рекомендации новых треков на основе прослушанной музыки. - Поиска треков по артисту, жанру, рейтингу. - Сортировки треков в плейлисте по различным критериям. - Анализа предпочтений пользователя для рекомендаций. 3) Создайте приложение, позволяющее пользователям управлять своей музыкальной библиотекой, создавать плейлисты и получать персональные рекомендации.
8	Управление расписанием занятий университета.

	1) Создайте следующие data-классы: - Преподаватель: имя, список дисциплин. - Студент: имя, группа, список курсов. - Аудитория: номер, вместимость, оборудование. - Занятие: дисциплина, преподаватель, аудитория, время, группа. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Планирования занятий с учётом расписания преподавателей, студентов и доступности аудиторий. - Проверки конфликтов в расписании. - Фильтрации расписания по преподавателю, группе, аудитории. - Поиска свободных аудиторий в заданный промежуток времени. - Анализа нагрузки на преподавателей и аудитории. 3) Создайте приложение, позволяющее администраторам составлять и редактировать расписание, а студентам и преподавателям – просматривать своё расписание.
9	Анализ данных социальных медиа. 1) Разработайте следующие data-классы: - Пользователь: имя, список друзей, список постов. - Пост: автор, содержание, дата публикации, список лайков и комментариев. - Комментарий: автор, содержание, дата. 2) Используя коллекции и лямбда-выражения, создайте функции для: - Создания постов и комментариев. - Лайков постов другими пользователями. - Добавления друзей. - Поиска наиболее популярных постов (по количеству лайков и комментариев). - Анализа активности пользователей (количество постов, комментариев). - Рекомендации друзей на основе общих знакомых. 3) Создайте приложение, которое собирает данные и предоставляет отчёты по активности пользователей, популярным темам и связям между пользователями.

3 Материально-техническое обеспечение работы

Аудитория для проведения лабораторных занятий должна быть укомплектована специализированной мебелью и индивидуальными компьютерами следующей минимальной комплектации:

– Процессор: не менее двух исполнительных ядер, совместимый с системой команд x86 и x64, с поддержкой аппаратной виртуализации.
– Оперативная память: не менее 8 Гб.
– Монитор: не менее 24" (дюймов) по диагонали.
– Наличие локальной сети со скоростью обмена не менее 1 Гб/сек.
– Наличие доступа в сеть Интернет со скоростью не менее 1 Мбит/сек.
– Наличие клавиатуры и манипулятора «мышь».

На компьютерах должно быть установлено следующее программное обеспечение:

– Операционная система: любая современная операционная система не ранее 2022 года выпуска допускающая установку JDK версии не ниже 21 и среды разработки программного обеспечения, поддерживающей язык программирования Kotlin.
– Среды разработки: IntelliJ IDEA Community Edition или Visual Studio Code.
– Среда исполнения: JDK версии не ниже 21.

4 Порядок выполнения и сдачи работы

Для выполнения лабораторной работы рекомендуется придерживаться следующего порядка выполнения:

1. Ознакомится с темой и целями лабораторной работы
2. Изучить теоретический материал
3. Подготовить рабочее окружение
4. Разработать программный код лабораторной работы в соответствие с заданием
5. Произвести проверку работоспособности, выполнить тестирование и отладку кода
6. Проанализировать полученные результаты и примененные в ходе решения подходы
7. Выполнить самооценку и рефлексию
8. Сдать лабораторную работу преподавателю и получить от него обратную связь

При сдаче студентом лабораторной работы основным отчетом выступает исходный код самостоятельно созданной в процессе выполнения работы программы.

Исходный код должен быть отформатирован согласно принятым для используемого языка программирования стандартам. Является желательным наличие в исходном коде комментариев, описывающих основные части программы и особенности их функционирования. В то же время студент должен быть готов объяснить работу программы в целом и каждую отдельную ее часть при полном отсутствии комментариев (например, они могут быть удалены преподавателем при сдаче работы).

Обязательным условием сдачи является умение студента самостоятельно восстановить любой участок исходного кода программы (но не более 20 строк подряд) после его удаления. Удаленный участок должен быть самостоятельно восстановлен обучающимся заново в присутствие преподавателя. При восстановлении удаленного участка кода запрещается использовать операции Undo (Отменить) текстовых редакторов или переписывание кода участка из других источников.

Исходный код сдаваемой программы должен быть представлен в электронном виде.

Сдача исходного кода ранее сдававшихся программ или программ, код которых выложен в сети Интернет, не допускается.

5 Контрольные вопросы к лабораторной работе

1. Что такое data-класс в Kotlin и какие преимущества он предоставляет по сравнению с обычными классами?
2. Что делают автоматически генерируемые методы equals(), hashCode() и toString() в data-классах?
3. Объясните, как использовать функцию copy() в data-классах и для чего она может быть полезна.
4. Как создать неизменяемый список в Kotlin и чем он отличается от изменяемого списка?
5. Какие стандартные коллекции предоставляет Kotlin и как выбрать подходящую для конкретной задачи?
6. Каковы основные отличия между списками (List), множествами (Set) и картами (Map) в Kotlin?
7. Что такое функция высшего порядка и каковы примеры ее использования в Kotlin?

8. Объясните, как работают лямбда-выражения в Kotlin и приведите пример их использования.
9. Как использовать метод filter для коллекций в Kotlin и в каких случаях он может быть полезен?
10. Что делает метод map в контексте коллекций и как он может изменить элементы коллекции?
11. Как использовать функции fold и reduce для агрегации данных в коллекциях?
12. Объясните, что такое анонимная функция и как она отличается от лямбда-выражения.
13. Как использовать метод forEach для итерации по элементам коллекции и чем он отличается от обычного цикла for?
14. В чем разница между функциями take и drop в работе с коллекциями?

6 Перечень использованных информационных ресурсов

1. Kotlin Docs | Kotlin Documentation [Электронный ресурс]. – URL: https://kotlinlang.org/docs/home.html
2. Kotlin | Изменяемые и неизменяемые коллекции [Электронный ресурс]. – URL: https://metanit.com/kotlin/tutorial/7.1.php
3. IntelliJ IDEA – the Leading Java and Kotlin IDE [Электронный ресурс]. – URL: https://www.jetbrains.com/idea/
4. Visual Studio Code - Code Editing. Redefined [Электронный ресурс]. – URL: https://code.visualstudio.com/

Редактор А.А. Литвинова

ЛР № 04779 от 18.05.01.	В набор	В печать
Объем 0,5 усл.п.л., уч.-изд.л.	Офсет.	Формат 60x84/16.
Бумага тип №3.	Заказ №	Тираж 75. Цена

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344010, г. Ростов-на-Дону, пл. Гагарина, 1.