ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 025053D20009B2A5B043583E315FEF31AD
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 14.10.2024 до 14.01.2026

УТВЕРЖДЕНА

Приказом Ректора АНО ВО
«Центральный университет»
Ивашкевич Е.В.
от «19» января 2024 г. № 0119.37

Рабочая программа дисциплины (модуля) «Java Core (Основы разработки на языке Java)» дополнительной профессиональной программы – программы профессиональной переподготовки «Академия data science»

Траектория: Машинное обучение

Москва
2024

Электронный документ

Содержание

1. Краткая характеристика дисциплины (модуля)

Изучение дисциплины (модуля) «Java Core (Основы разработки на языке Java)» помогает слушателям овладеть знаниями Java, которые являются базой для понимания объектно-ориентированного программирования и разработки программного обеспечения на одном из самых популярных и востребованных языков в индустрии. Освоение этих основ позволяет создавать масштабируемые приложения и обеспечивает конкурентоспособность на рынке труда.

Цель изучения дисциплины (модуля): формирование у слушателей фундаментальных концепций и синтаксис языка программирования Java для разработки эффективных и надежных приложений.

Задачи изучения дисциплины (модуля):

— формирование знания экосистемы и синтаксиса Java, базовые и управляющие конструкции;

— формирование знания понятия классов и объектов, наследование, полиморфизм и инкапсуляция, интерфейсы и абстрактные классы;

— формирование знания коллекций и структуры данных; сравнение и хэширование (equals(), hashCode(), Comparable), лямбда выражения и функциональные обобщения;

— формирование знания основ сетевого программирования;

— формирование умения найти ошибку в коде и исправить ошибку в окружении и в логике;

— формирование умения настраивать окружение экосистемы Java;

— формирование умения обрабатывать исключения с использованием try-catch

— формирование умения создавать объекты Optional, а также совместно использовать Optional и stream API

— формирование навыка правильно ставить задачи для получения необходимого результата.

— формирование навыка писать читаемый и поддерживаемый код;

— формирование навыка применять принципы ООП для создания приложений;

— формирование навыка уверенно пользоваться IDE.

2. Тематический план

№ п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары (Практические занятия)			
1	Основы языка	3	3		10	Домашнее задание
2	ООП	3	3		10	Домашнее задание
3	Исключения	3	4		10	Домашнее задание
4	Generics и коллекции	4	4		10	Домашнее задание
5	Функциональное программирование	4	4		10	Домашнее задание
6	Аннотации и Reflection	4	4		10	Домашнее задание
7	Сборка и управление зависимостями в Java с использованием Gradle	4	4		10	Домашнее задание
8	Основы сетевого программирования	4	4		12	Домашнее задание
9	Основы многопоточности: потоки, синхронизация и блокировки, ThreadPool и Executor Service	4	4		12	Домашнее задание
	Зачет			***4***		***Устный опрос***
Итого:		***33***	***34***	***4***	***94***	
Объем дисциплины (модуля) (в ак. ч.)		***165***				

3. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Основы языка	История языка, главные отличия от остальных. Консольный ввод-вывод. Примитивные типы и операции над ними. Циклы, условные операторы, switch. Строки, массивы, перечисления. Функции, выбор версии. Области памяти и работа с объектами. Static переменные и память
2	ООП	Классы: поля и методы, модификаторы доступа public, private. Конструкторы и this. Static-методы и -поля, final-поля и переменные. Import классов, пакеты, package private modifier. Внутренние и анонимные классы. Наследование, переопределение методов и вызов нужной версии, super. final class, final method. Абстрактные методы и классы. Интерфейсы
3	Исключения	Способы обработки ошибок. checked- и unchecked-исключения, иерархия, try-catch, finally block, try-with-resources
4	Generics и коллекции	Объявление generic-классов, интерфейсов и методов. Ограничения типов, PECS. type erasure, wildcards. Обзор коллекций: динамический массив, список, множество, словарь, очередь. Итератор. Сравнение и хэширование: equals(), hashCode(), Comparable

5	Функциональное программирование	Лямбда-выражения, интерфейсы java.util.function, ссылки на методы. Optional: создание объектов и использование. Stream API: создание потоков, промежуточные и терминальные операции. Совместное использование Optional и Stream API
6	Аннотации и Reflection	Обзор встроенных в язык аннотаций. Мета-аннотации @Target, @Retention, @Inherited. Создание пользовательских аннотаций, параметризация. Reflection: концепция и принцип работы. Доступ к компонентам класса, создание экземпляров и манипуляция объектами. Производительность и безопасность
7	Сборка и управление зависимостями в Java с использованием Gradle	Ключевые понятия и структура gradle-проекта, конфигурации, базовые задачи сборки. Управление зависимостями и использование Maven Central. Создание и настройка JAR и распространение модулей. Создание собственных gradle-плагинов
8	Основы сетевого программирования	Обзор компонентов сети и протоколов, TCP/IP. Клиент-серверная архитектура. Сокеты и методы создания соединения. Обработка исключений в сетевых приложениях
9	Основы многопоточности: потоки, синхронизация и блокировки, ThreadPool и Executor Service	Жизненный цикл потока, Thread, Runnable. Проблемы многопоточности и понятие гонки данных. Синхронизация и блокировка: synchronized, locks. Асинхронность и Future. thread pool и идея повторного использования потоков. Работа с ExecutorService для управления потоками

4. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый слушатель в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Слушателям обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Кубенский, А. А. Функциональное программирование : учебник и практикум для вузов / А. А. Кубенский. — Москва : Издательство Юрайт, 2025. — 348 с. — (Высшее образование). — ISBN 978-5-9916-9242-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/561074.

Дополнительная литература:

1. Зыков, С. В. Программирование : учебник и практикум для вузов / С. В. Зыков. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 285 с. — (Высшее образование). — ISBN 978-5-534-16031-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/560815.

5. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;

— компьютерной техникой;

— механическими калькуляторами;

— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1	Катастрофы, стихийные бедствия, аварии, эпидемии. Солнечная и геомагнитная активность. /ежедневный обзор	http://www.disasters.chat.ru
2	Каталог по безопасности жизнедеятельности	http://www.eun.chat.ru
3	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
4	База данных для IT-специалистов	https://habr.com
5	База данных ScienceDirect	https://www.sciencedirect.com
6	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
7	Федеральный портал «Российское образование"	https://www.edu.ru/
8	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
9	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
10	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/
11	Сайт различных плагинов	https://maven.apache.org/plugins/
12	Maven central repository - хранилище библиотек и фреймворков	https://mvnrepository.com/repos/central

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное

Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

6. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Java Core (Основы разработки на языке Java)» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, домашние задания, устный опрос, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Семинар — это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где слушатели активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Домашнее задание – набор заданий по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал. Использовать различные источники информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

Устный опрос – форма проверки знаний, при которой слушатель отвечает на вопросы преподавателя в устной форме, демонстрируя свои знания и понимание темы.

Для успешной подготовки к устному опросу рекомендуется тщательно изучить материалы по теме, включая лекции и учебники. Создайте список ключевых вопросов и ответов, а также практикуйтесь в их проговаривании вслух, чтобы улучшить уверенность и четкость изложения. Полезно также обсудить темы с однокурсниками, чтобы получить разные точки зрения и углубить понимание материала. Наконец, не забывайте о важности хорошего самочувствия и спокойствия в день опроса, что поможет сосредоточиться на ответах.

Самостоятельная работа – работа слушателей, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы слушатели взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи слушателя включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Оценивание уровня учебных достижений обучающихся по дисциплине (модулю) осуществляется в виде текущего контроля успеваемости.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачета***.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	**Пятибалльная оценка**	**Оценка за зачет**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Зачтено	Слушатель полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину (модуль). Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Слушатель хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	Зачтено	
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Слушатель обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи.
6	Хорошо	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Слушатель хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
5	Удовлетворительно	Зачтено	Слушатель обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Слушатель способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	Зачтено	
3	Не сдан	Не зачтено	Слушатель не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Java Core (Основы разработки на языке Java)» оценивается следующим образом:

Активность	Вес	Описание
Домашние задания	70%	Оцениваются по критериям. Можно набрать максимум 10 баллов за каждое из заданий.
Устный опрос (зачет)	30%	Зачёт с оценкой состоит из двух частей: 1. Устный ответ на три вопроса из разных тем, выбранных случайным образом. 2. Обсуждение фрагмента кода.

Формула расчёта итоговой оценки по дисциплине (модулю) «Java Core (Основы разработки на языке Java)»: «0,7 × среднее за домашние задания + 0,3 × устный опрос».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные домашние задания

Домашнее задание по лекциям 1 и 2: Калькулятор

Необходимо реализовать консольное приложение, которое принимает на вход строку с числами и арифметическими операциями и возвращает в стандартный вывод результат вычисления выражения или соответствующее сообщение об ошибке в случае некорректной входной строки

Разрешенные операции: `+, -, *, /, (, )`. Приоритет математический: сначала скобки, потом `*, /`, в конце `+, -`.

```
java Calculator "1 + (2 - 3) * 2"
```

`-1`

Для решения рекомендуется использовать `java.util.ArrayDeque` в качестве стека для операций и операндов.

Результат решения задачи — файл Calculator.java, содержащий класс Calculator с реализованным методом main

Формат ввода

Одна строка с числами и арифметическими операциями, все последующие игнорировать.

Разрешенные символы: — цифры [0, 9] — точка `.` для отделения целой части от дробной — операции: `+, -, *, /, (, )` — пробелы (могут отсутствовать)

Все остальные символы стоит считать некорректными.

Формат вывода

Одно число — результат вычисления выражения. Для вещественного результата вывести максимум 2 символа после точки, например, `2.34` Для отрицательного: минус `-`, затем модуль числа без пробелов

Критерии оценивания

Выполненное задание максимально оценивается 10-ю баллами при пройденном код-ревью и выполненными условиями, описанными в задании. За задание, выполненное без поддержания скобок - максимум 7 баллов, без валидации исходной строки — максимум 8 баллов, без скобок и валидации — максимум 5 баллов.

Домашнее задание

Геометрия

Напишите иерархию классов для работы с геометрическими фигурами на плоскости.

- `record Point` — точка на плоскости. Точку можно задать двумя числами типа double. Точки можно сравнивать с помощью метода `equals(Point another)`.
- Класс `Line` — прямая. Прямую можно задать двумя точками, можно двумя числами (угловой коэффициент и сдвиг), можно точкой и числом (угловой коэффициент). Прямые можно сравнивать с помощью метода `equals(Line another)`. Прямые можно персекать с помощью статического метода `Point intersect(Line first, Line second)`. в случае невозможности это сделать нужно вернуть `Point.EMPTY` — статическое поле `Point`, пустая точка, которая не представляет ни одну точку на плоскости
- Интерфейс `Shape` — фигура.
- Класс `Polygon` — многоугольник. Многоугольник — частный случай фигуры. У многоугольника можно спросить:
    - `int verticeCount()` — количество вершин
    - `Point[] vertices()` — сами вершины без возможности изменения.
    - `boolean isConvex()` — выпуклый ли

  Можно сконструировать многоугольник из массива точек-вершин в порядке обхода. Можно сконструировать многоугольник из точек, передаваемых в качестве параметров через запятую (т.е. неуказанное число аргументов). Для простоты будем считать, что многоугольники с самопересечениями никогда не возникают (гарантируется, что в тестах таковые будут отсутствовать). Кроме того, считаем, что три последовательных вершины многоугольника никогда не лежат на одной прямой.

- Класс `Ellipse` — эллипс. Эллипс — частный случай фигуры. У эллипса можно спросить
  - `record Focuses(Point left, Point right) focuses()` — его фокусы
  - `record Directrices(Line first, Line second) directrices()` — пару его директрис
  - `double eccentricity()` — его эксцентриситет
  - `Point center()` — его центр

  Эллипс можно сконструировать из двух точек и double (два фокуса и сумма расстояний от эллипса до них)

- Класс `Circle` — круг. Круг — частный случай эллипса. У круга можно спросить:
  - `double radius()` — радиус

  Круг можно задать точкой и числом (центр и радиус).

- Класс `Rectangle` — прямоугольник. Прямоугольник — частный случай многоугольника. У прямоугольника можно спросить:
  - `Point center()` — его центр
  - `record Diagonals(Line first, Line second) diagonals()` — пару его диагоналей

  Прямоугольник можно сконструировать по двум точкам (его противоположным вершинам) и числу (отношению смежных сторон), причем из двух таких прямоугольников выбирается тот, у которого более короткая сторона расположена по левую сторону от диагонали, если смотреть от первой заданной точки в направлении второй.

- Класс `Square` — квадрат. Квадрат — частный случай прямоугольника. У квадрата можно спросить:
  - `Circle circumscribedCircle()` — описанная окружность
  - `Circle inscribedCircle()` — вписанная окружность

  Квадрат можно задать двумя точками — противоположными вершинами.

- Класс `Triangle` — треугольник. Треугольник — частный случай многоугольника. У треугольника можно спросить:
  - `Circle circumscribedCircle()` — описанная окружность
  - `Circle inscribedCircle()` — вписанная окружность

У любой фигуры можно спросить:

- `double perimeter()` — периметр
- `double area()` — площадь
- `boolean equals(Shape another)` — совпадает ли эта фигура с другой как множество точек. (В частности, треугольник ABC равен треугольнику BCA.)
- `boolean isCongruentTo(Shape another)` — равна ли эта фигура другой в геометрическом смысле, то есть можно ли совместить эти фигуры движением плоскости. Движение — это отображение плоскости на себя, сохраняющее расстояния.

- `boolean isSimilarTo(Shape another)` — подобна ли эта фигура другой, то есть можно ли перевести одну фигуру в другую преобразованием подобия. (Определение преобразования подобия, кто не знает, можно посмотреть в Википедии.)
- `boolean containsPoint(Point point)` — находится ли точка внутри фигуры.

Фигуры не обязаны иметь одинаковый тип, чтобы считаться равными, конгруэнтными или подобными! Любую фигуру должно быть можно сравнить с любой другой и получить правильный ответ, независимо от настоящих типов этих фигур!

С любой фигурой можно сделать:

- `rotate(Point center, double angle)` — поворот на угол (в градусах, против часовой стрелки) относительно точки
- `reflect(Point center)` — симметрию относительно точки
- `reflect(Line axis)` — симметрию относительно прямой
- `scale(Point center, double coefficient)` — гомотетию с коэффициентом `coefficient` и центром `center`

Ваши файлы должны находится в пакете `hometask.geometry`. В пакете не должно быть класса с методом `main`. Для вычисления математических функций стоит использовать методы класса `java.lang.Math`. В качестве jdk рекомендуется использовать версию 21.

Домашнее задание

Список с пропусками

Напишите класс — реализацию множества на основе списка с пропусками

Список с пропусками (Skip list) — это вероятностная структура данных, позволяющая вставлять, удалять и искать элементы в среднем за

O(logn)O(log{n})O(logn)

. Представляет из себя несколько уровней, каждый из которых является отсортированным односвязным списком.

Внутреннее устройство

Нижний уровень содержит все элементы в отсортированном односвязном списке, последующие уровни содержат также отсортированную цепочку элементов, но некоторые из нижнего уровня могут отсутствовать. Элемент, находящийся на уровне `i`, содержится в уровне `i + 1` с некоторой фиксированной вероятностью (как правило это `1/4`). Количество уровней ограничено максимальным числом.

Поиск элемента

Поиск начинается с первого элемента верхнего уровня. Пока текущий элемент меньше искомого, выбирается следующий элемент того же уровня. Если найден равный элемент, поиск завершается успешно. Если дошли до конца списка, то элемент не найден. Если же встретился элемент, больший искомого, то выбирается предыдущий (последний из меньших) и процедура поиска повторяется с него уже на следующем, низшем уровне.

Вставка

Сначала выбирается существующий элемент на нижнем уровне, после которого будет добавляться новый (наибольший из всех меньших или ближайший слева). Элемент вставляется на нижний уровень, после "подкидывается монетка" и решается, добавлять ли этот новый элемент на предыдущий уровень. Если результат положительный, то после вставки монетка подкидывается вновь для определения, добавлять ли уже на уровень `n -`

2. И так до тех пор, пока не выпадет отрицательный результат или элемент добавится на все возможные уровни.

Удаление

После успешного нахождения элемента на нижнем уровне он удаляется со всех уровней, начиная с верхнего.

Детали реализации

Класс `SkipListSet` должен быть обобщенным, имплементировать интерфейс `java.util.NavigableSet`и расширять `java.util.AbstractSet`.

Должны быть реализованы методы:

```
int size();
boolean isEmpty();
boolean contains(Object o);

// Добавляет элемент в set, если его там не было, и возвращает `true`,
иначе `false`
boolean add(E e);

// Удаляет элемент из set, если он содержался внутри, и возвращает
`true`, иначе `false`
boolean remove(Object o);

// Очищает все содержимое
void clear();

// Возвращает Comparator, с которым был создан set, или null
Comparator<? super E> comparator();

// Возвращает первый (минимальный) элемент или null, если set пустой
E first();

// Удаляет первый (минимальный) элемент и возвращает его, или null,
если set пустой
E pollFirst();

// Возвращает последний (максимальный) элемент или null, если set
пустой
E last();

// Удаляет последний (максимальный) элемент и возвращает его, или
null, если set пустой
E pollLast();

// Наибольший элемент, который строго меньше искомого, или null
E lower(E e);

// Наибольший элемент, который меньше или равен искомому, или null
E floor(E e);

// Наименьший элемент, который больше или равен искомому, или null
E ceiling(E e);

// Наименьший элемент, который строго больше искомого, или null
E higher(E e);
```

```
// Восходящий итератор. В итераторе должны быть реализованы методы
`hasNext`, `next` и `remove` согласно описанию в javadoc
Iterator<E> iterator();

// Нисходящий итератор
Iterator<E> descendingIterator();

// Set, отсортированный в обратном порядке. Изменения в этом set
должны приводить к соответствующим изменениям
// оригинального set и наоборот
NavigableSet<E> descendingSet();

// Часть set с `fromElement` по `toElement`. Изменения в этом set
должны приводить к соответствующим изменениям
// оригинального set и наоборот
NavigableSet<E> subSet(E fromElement, boolean fromInclusive, E
toElement, boolean toInclusive);
SortedSet<E> subSet(E fromElementInclusive, E toElementExclusive);

// Часть set с начала по `toElement`. Изменения в этом set должны
приводить к соответствующим изменениям
// оригинального set и наоборот
NavigableSet<E> headSet(E toElement, boolean inclusive);
SortedSet<E> headSet(E toElementExclusive);

// Часть set, начиная с `fromElement`. Изменения в этом set должны
приводить к соответствующим изменениям
// оригинального set и наоборот
NavigableSet<E> tailSet(E fromElement, boolean inclusive);
SortedSet<E> tailSet(E fromElementInclusive);
```

Более подробное описание методов можно посмотреть в javadoc интерфейса, стоит следовать контрактам, которые там описаны

Класс можно сконструировать с компаратором; от любой коллекции; без параметров, тогда (как и в случае с коллекцией) параметризованный тип должен имплементировать `Comparable<E>`, а если параметризовать типом без `Comparable<E>`, то методы `add`, `remove`, `contains` и другие, принимающие обект или коллекцию параметризованного типа `E`, должны кидать `ClassCastException`;

В качестве вероятности используйте число в промежутке `(0, 1/2]`, можно взять `1/4`. Для максимального количества уровней достаточно взять число 32.

Ваши файлы должны находиться в пакете `hometask.collections`. В пакете должен быть один класс `SkipListSet`. Для генерации случайных чисел можно использовать класс `java.util.Random` В качестве jdk рекомендуется использовать версию 21.

Примерные задания к устному опросу

1. Основы языка

— Примитивные типы данных и операции над ними. Приведение типов

— Функции (методы), перегрузка, выбор версии.

2. Работа с объектами

— Основные области памяти в java (стек, куча). Ошибки, связанные с каждой из областей; что к ним приводит.

— Объекты: создание и удаление. Методы Object, способы сравнить обекты.

3. ООП:

— Классы: что это такое, зачем нужны, из чего состоят, что с ними можно делать. this. Модификатор static.

— Типы классов: локальные, nested inner, nested static. Record. Enum.

4. Наследование

— Интерфейсы, абстрактные и анонимные классы: что это такое, зачем нужны; их отличия.

— 4 модификаторы доступа. Модификатор final.

— Переопределение методов, выбор версии. Ключевое слово super.

— Оператор instanceof, приведение типов.

5. Исключения

— Способы обработки ошибок, best practices. Типы исключений, иерархия.

— Блоки try-catch, finally, try-with-resources

6. Generics

— Обобщенные классы: создание, использование параметра, инициализация объекта.

— Обобщенные методы: объявление и вызов. Наследование обобщенных классов.

— Wildcards, верхние и нижние границы. PECS. Bounded type.

— Type erasure: что это, какие накладывает ограничения. Heap pollution.

7. Коллекции

— Иерархии коллекций: интерфейсы, абстрактные классы и реализации

— Iterable, iterator, for(var s: iterable), ConcurrentModificationException. Comparable, Comparator.

— Ассоциативный массив. Map, устройство HashMap и TreeMap, ограничение на ключи

8. Лямбда выражения и функциональные обобщения

— @FunctionalInterface: требования и ограничения. Примеры из jdk, зачем нужен каждый

— Лямбда-выражения. Разные варианты синтаксиса. Захват окружения.

— Ссылки на методы. Особенности и ограничения лямбда выражений и ссылок на методы.

9. optional & stream

— Optional: для чего нужен, создание, методы. Best practices.

— Stream API: для чего нужен, промежуточные и терминальные операции. Best practices.

— Коллекторы: из чего состоят, примеры, комбинации разных. Создание своего коллектора.

10. Аннотации

— Что такое аннотации и зачем они нужны. Как объявить и использовать. Где могут быть использованы.

— @Target, @Retention. Параметры аннотаций.

11. Reflection

— Что это такое, для чего используется. Что можно извлечь про дженерики.

— Методы для доступа к конструкторам, полям, методам и аннотациям.

— Инициализация поля, вызов метода через reflection.

12. Gradle

— Зачем нужна система сборки. Файлы проекта gradle. Plugins, tasks.

— Типы зависимостей. Распространение библиотек. Jar, Manifest.

13. IO & tests:

— InputStream, OutputStream, Reader, Writer: для чего нужен каждый, как использовать.

— Что позволяет делать java.io.File. Для чего нужны Path, Files, Paths из java.nio;

отличие от File

— Явное и автоматическое управление ресурсами

— tests: best practices

14. jvm

— ClassLoader: что это такое, из чего состоит, зачем нужны разные? Инициализация класса.

Приведение типов классов, загруженными разными classLoader

— Сборка мусора: типы сборки мусора их особенности, гипотеза поколений, регионы, примеры разных GC