ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«24» июня 2025 г.
Протокол № 2

Рабочая программа дисциплины (модуля)
«Java Core (Основы разработки на языке Java)»

Направление подготовки: *02.04.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Backend-разработка*

Квалификация (степень) выпускника: *магистр*

Форма обучения: *очная*

Срок освоения программы: *2 года*

Год набора: *2025*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

Электронный документ

Москва
2025

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Java Core (Основы разработки на языке Java)» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – магистратура по специальности 02.04.01 Математика и компьютерные науки, профиль Backend-разработка, утвержденный приказом Министерства науки и высшего образования Российской Федерации № 810 от 23.08.2017 года.

Изучение дисциплины (модуля) «Java Core (Основы разработки на языке Java)» помогает студентам 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.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки магистратуры по направлению 02.04.01 Математика и компьютерные науки, профиль Backend-разработка и входит в обязательную часть Блока 1.

Дисциплина (модуль) изучается на 1 курсе в 1 семестре.

Цель изучения дисциплины (модуля): формирование у студентов фундаментальных знаний и практических навыков программирования на языке Java для разработки эффективных и поддерживаемых приложений.

Задачи изучения дисциплины (модуля):

- освоить настройку и использование инструментов разработки для создания Java-программ;
- научиться анализировать и исправлять ошибки в программном коде;
- развить умение понимать и интерпретировать чужой код на Java;
- приобрести навыки написания структурированного и понятного кода;
- овладеть методами применения современных возможностей языка для решения практических задач.

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

- синтаксис языка Java, базовые и управляющие конструкции;
- основы ООП: понятия классов и объектов, наследование, полиморфизм и инкапсуляция, интерфейсы и абстрактные классы;
- принципы работы с исключениями в Java;
- основные коллекции и структуры данных в Java, принципы сравнения объектов;
- принципы работы с дженериками, стримами, лямбда-выражениями и функциональными интерфейсами;
- принципы работы аннотаций и рефлексии в Java.

уметь:

- устанавливать и настраивать окружение Java для разработки;
- находить ошибки в коде и исправлять их;
- уверенно пользоваться IDE;
- читать чужой код на Java и понимать, что он делает;
- писать читаемый и поддерживаемый код на Java.

владеть:

- навыками работы с исключениями, их обработки с использованием try-catch;
- навыками работы с данными при помощи коллекций и Stream API;

— навыками создания приложений, соответствующих принципам ООП.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
ОПК-3.	Способен самостоятельно создавать прикладные программные средства на основе современных информационных технологий и сетевых ресурсов, в том числе отечественного производства	ОПК-3.1.	Знает основные принципы программирования, архитектуры программного обеспечения и современные языки программирования, а также особенности отечественных информационных технологий и сетевых ресурсов
		ОПК-3.2.	Умеет разрабатывать прикладные программные средства, используя современные инструменты и технологии, а также интегрировать их с сетевыми ресурсами для решения конкретных задач
		ОПК-3.3.	Имеет практический опыт разработки программных средств, используемых при построении математических моделей в естественных науках
ПК-6.	Способен разрабатывать программное обеспечение для решения прикладных задач в сфере профессиональной деятельности	ПК-6.1.	Знает основные языки программирования, методы разработки программного обеспечения, а также принципы проектирования и архитектуры программных систем, применяемых в конкретной предметной области
		ПК-6.2.	Умеет анализировать прикладные задачи, разрабатывать алгоритмы и реализовывать их в виде программного обеспечения, используя современные инструменты и технологии, а также проводить тестирование и отладку созданных решений
		ПК-6.3.	Имеет практический опыт разработки программного обеспечения в рамках реальных проектов, включая участие в командах, где были успешно реализованы решения для конкретных прикладных задач в сфере профессиональной деятельности

3. 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 Java	2	2		8	Домашнее задание
2	Работа с объектами	2	2		8	Домашнее задание
3	Объектно-ориентированное программирование	2	2		8	Домашнее задание
4	Наследование	2	2		8	Домашнее задание
5	Исключения	2	2		8	Домашнее задание
6	Дженерики	2	2		8	Домашнее задание
7	Коллекции	2	2		8	Домашнее задание
8	Лямбда-выражения и функциональные интерфейсы	2	2		8	Домашнее задание
9	Optional и Stream API	2	2		8	Домашнее задание
10	Аннотации	2	2		9	Домашнее задание
11	Reflection	2	2		9	Домашнее задание
12	Сборка и управление зависимостями в Java	2	2		9	Домашнее задание
13	Ввод-вывод и работа с файлами	2	2		9	Домашнее задание
14	Модульное тестирование	2	2		9	Домашнее задание
15	Внутреннее устройство JVM	2	2		9	Домашнее задание
	Экзамен			***4***		
	Итого:	***30***	***30***	***4***	***126***	
Объем дисциплины (модуля) (в ак. ч.)		***190***				
Объем дисциплины (модуля) (в зач. ед.)		***5***				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Основы языка Java	Стек протоколов. Прикладной уровень
2	Работа с объектами	Прикладной и транспортный уровни
3	Объектно-ориентированное программирование	Транспортный уровень
4	Наследование	Сетевой уровень
5	Исключения	Канальный и физический уровни
6	Дженерики	Протоколы маршрутизации
7	Коллекции	Организация глобальных сетей
8	Лямбда-выражения и функциональные интерфейсы	Беспроводные и мобильные сети
9	Optional и Stream API	Отказоустойчивые и виртуальные сети

10	Аннотации	Виртуальные частные сети
11	Reflection	Шифрование данных
12	Сборка и управление зависимостями в Java	Аутентификация и авторизация
13	Ввод-вывод и работа с файлами	Анализ и фильтрация траффика
14	Модульное тестирование	Атаки на TCP/IP
15	Внутреннее устройство JVM	Атаки на DNS

5. Учебно-методическое обеспечение

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

Каждый студент 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.

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

Основная литература:

1. Эккель, Б. Философия Java : практическое руководство / Б. Эккель. - 4-е полное изд. - Санкт-Петербург : Питер, 2019. - 1168 с. - (Серия «Классика computer science»). - ISBN 978-5-4461-1107-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1856792.

2. Васильев, А. Н. Java. Объектно-ориентированное программирование : учебное пособие / А. Н. Васильев. - Санкт-Петербург : Питер, 2021. - 395 с. - ISBN 978-5-4461-9365-3. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2139140.

Дополнительная литература:

1. Чан, Д. Java: быстрый старт : практическое руководство / Д. Чан. - Санкт-Петербург : Питер, 2021. - 272 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-1801-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1739592.

2. Васильев, А. Java для всех : практическое руководство / А. Васильев. - Санкт-Петербург : Питер, 2020. - 512 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-1382-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1756101.

6. 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 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

7. 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, 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 студенты 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.

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

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента 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)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Java Core (Основы разработки на языке Java)».

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

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***экзамена***, при этом проводится оценка компетенций, сформированных по дисциплине.

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

Десятибалльная оценка	**Пятибалльная оценка**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Студент 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. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	
8	Отлично	
7	Хорошо	Студент 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. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
6	Хорошо	
5	Удовлетворительно	Студент 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.
4	Удовлетворительно	

Десятибалльная оценка	Пятибалльная оценка	Общая характеристика результата обучения по дисциплине (модулю)
		Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
3	Не сдан	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	
1	Не сдан	

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.
Экзамен	30%	Экзамен 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 × экзамен».

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.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1.	Какое из следующих утверждений о перегрузке функций в Java является истинным? A) Перегрузка функций позволяет существовать нескольким методам с одинаковым именем в одном классе, при условии, что у них разные типы параметров или разное количество параметров. B) Перегрузка функций возможна только для методов, имеющих одинаковый тип возвращаемого значения. C) Перегрузка функций может быть достигнута только изменением имени метода. D) Перегрузка функций не поддерживается в Java; вместо этого Java использует переопределение методов.	A	ПК-6
2.	Какой модификатор доступа делает переменную или метод доступным только внутри класса?	Private/ private	ПК-6
3.	Соотнесите ключевые слова Java с их назначением. ***Ответ напиши в виде пар цифра-буква без пробелов и других символов, например, 1A2B3C4D*** 1) final 2) static 3) super	1B2D3A4C/BDAC	ПК-6

	4) this A) используется для обращения к родительскому классу B) используется для обозначения неизменяемых переменных или методов C) используется для обращения к текущему экземпляру класса D) применяется для обозначения методов и полей, принадлежащих классу, а не объекту		
4.	Что выведет в стандартный вывод следующий отрывок кода? ```java String s = "abc"; String t = s; s += "d"; System.out.println(t); ```	abc	ОПК-3
5.	Как объявить массив целых чисел, содержащий 42 элемента? В ответе укажите то, что нужно написать вместо ... в строчке кода: `int[] array = ...;`	new int[42]	ПК-6
6.	Расположите этапы обработки исключений в Java в правильном порядке. ***Ответ напишите в виде последовательности букв без пробелов и других символов, например, ABCD*** A) Выполняется код в блоке finally (если он есть). B) Проверяется код в блоке try на наличие исключений. C) Если возникает исключение, выполнение переходит в соответствующий catch-блок (если он есть). D)Если исключение не обработано, оно передается выше по стеку.	BCAD	ПК-6
7.	Какой метод Stream API используется для преобразования элементов стрима в другой тип? Напишите название этого метода без других символов	Map/map	ПК-6
8.	Что из следующего верно относительно лямбда-выражений в Java? A) Лямбда-выражения могут быть переданы в качестве аргументов методам B) Тип лямбда-выражения может быть выведен компилятором C) Для того, чтобы захватить переменную в лямбда-выражение, она должна быть final или effectively final D) Любой анонимный класс можно заменить лямбда-выражением	AC	ОПК-3

9.	Какие из следующих утверждений об аннотациях в Java являются верными? A) Аннотации могут использоваться для предоставления метаданных о классе, методах и полях. B) Аннотация @Retention определяет, как долго аннотация будет сохраняться (например, только в исходном коде, в байт-коде или во время выполнения). C) Аннотация @Target определяет, какие параметры могут быть переданы аннотации. D) Аннотации не могут применяться к другим аннотациям.	AB	ОПК-3
10.	Какая коллекция в Java реализует интерфейс List и позволяет доступ к элементам по индексу? Напишите название класса без пробела	ArrayList/ Arraylist/ arrayList/ arraylist	ОПК-3
11.	Какой метод используется для проверки, пуста ли коллекция в Java? Напишите название метода без других символов и без пробела	isEmpty/ istmpty/ IsEmpty/ Isempty	ОПК-3
12.	Какой оператор используется для наследования классов в Java?	Extends/ extends	ОПК-3
13.	Какой модификатор нужно применить к классу, чтобы нельзя было создать его экземпляр?	Abstract / abstract	ОПК-3