Для всех задач необходимо дополнительно создавать класс Application, который будет содержать статический метод main.

**Задача 1**

Создать класс MyArrayList<T>, который будет дополнять класс ArrayList<T>. Класс должен при вызове методов выводить результат операции на экран. Реализация методов должна оставаться такой же, как и у ArrayList, со следующими изменениями:

Метод get(int index) в случае, если индекс получаемого элемента находится за границами списка, должен возвращать null, а не выбрасывать исключение IndexOutOfBoundsException.

Метод add(int index, T element) в таком же случае должен добавлять элемент на позицию 0.

|  |  |
| --- | --- |
| Дополняемый метод | Вывод на экран |
| size() | Размер спика: <размер списка> |
| add(T t) | Добавлен объект <объект> в список |
| contains(Object o) | Объект содержится в списке ИЛИ Объект не содержится в списке |
| get(int index) | Получен объект <объект> из списка |
| add(int index, T element) | Добавлен объект <объект> в список на позицию <позиция>. |

В методе main необходимо протестировать реализацию методов описанного класса MyArrayList<T>

**Задача 2**

Необходимо создать класс-обертку ThreeHashMap<K, V1, V2> на основе класса HashMap<K, V>. Обертка должна реализовывать мапу, где каждому ключу соответствует 2 generic-значения и реализовывать следующие методы:

|  |  |
| --- | --- |
| Метод | Реализация метода |
| int size() | Возвращает размер содержимого, равный количеству хранимых троек. |
| boolean isEmpty() | Возвращает true, если мапа пустая и false, если есть тройки значений |
| boolean containsKey() | Возвращает true, если такой ключ представлен в мапе |
| boolean containsValue1(Object value) | Возвращает true, если среди значений V1 есть переданное |
| boolean containsValue2(Object value) | Возвращает true, если среди значений V2 есть переданное |
| Object getFirst(Object key) | Возвращает значение V1 по ключу |
| Object getSecond(Object key) | Возвращает значение V2 по ключу |
| void remove(Object key) | Удаляет значения V1, V2 по ключу |
| Set keySet() | Возвращает множество ключей K |
| Collection values1() | Возвращает множество значений V1 |
| Collection values2() | Возвращает множество значений V2 |
| void putAll(Map map1, Map map2) | Добавляет тройки в содержимое |
| void put(Object value1, Object value2) | Добавляет тройку в содержимое |
| clear() | Очищает содержимое |

Создать класс Person с полями, определяющими фамилию и имя человека

Создать класс Passport с полями, определяющими серию и номер паспорта.

В конструкторе Person и Passport реализовать присвоение случайных полей создаваемому объекту.

В методе main необходимо сгенерировать 10 человек и 10 паспортов. Создать ThreeHashMap<Integer, Person, Passport> и заполнить его сгенерированными данными. Первое значение типа Integer должно быть порядковым номером при заполнении структуры. Создать дополнительно Map<Integer, Person> persons и Map<Integer, Passport> passports на 20 случайных элементов каждый. При этом в первой мапе Integer является порядковым номером, начиная с 10, а во второй – начиная с 20. Проверить корректность метода putAll(persons, passports)

**Задача 3**

Дано: Имеется шахматное поле 8х8 (A-H, 1-8). Необходимо создать класс Player, являющийся перечислением возможных цветов игрока (черный и белый), и класс Position, содержащий два поля: положение по горизонтали и по вертикали. Реализовать базовый класс ChessFigure, описывающий фигуру, с методом getAvailableMoves(), который возвращает множество Position, на который может совершить свой ход фигура. Создать отдельный пакет Figures, в который поместить все фигуры. Создать пакет chess, в который поместить классы Player, Position и остальные необходимые для описания фигуры на поле.

Создать классы шахматных фигур King, Queen, Rook, Knight, Bishop, Pawn (Король, ферзь, ладья, конь, слон, пешка. Используя полиморфизм, переопределить метод getAvailableMoves() для каждой фигуры, чтобы метод возвращал возможные ходы для данной фигуры в зависимости от цвета игрока (для пешек) и позиции фигуры в соответствии с шахматными правилами.

В классе Application создать метод printMoves, который принимает множество возможных ходов и выводит их на экран в формате:

<Тип фигуры> (<Текущая позиция>): <Позиция1>, <Позиция2>, …, <Позиция N>

**Задача 4**

Необходимо создать базовый класс BaseCredit, описывающий кредит: поля amount, rate, duration (сумма, процентная ставка в год, длительность в месяцах). Реализовать в классе метод getMonthPayments, возвращающий список, который является графиком платежей. Реализовать в классе метод calculateOverpayment, считающий переплату по кредиту. В качестве типов данных для чисел с плавающей точкой использовать тип double.

Создать классы-наследники AnnuityCredit и DifferentiateCredit. Применяя полиморфизм, реализовать метод getMonthPayments следующим образом: AnnuityCredit должен считать график для аннуитетных платежей. DifferentiateCredit должен считать график для дифференцированных платежей. Метод базового класса должен игнорировать процентную ставку и считать график платежей беспроцентного кредита.

В методе main создать три кредита, каждый из которых должен иметь тип BaseCredit, а реализацию BaseCredit, DifferentiateCredit и AnnuityCredit соответственно. Метод main должен выводить в консоль строки в формате:

<№ месяца>: <Беспроцентный платеж>, <Дифф-ый платеж>, <Аннуитетный платеж>

При выводе данных необходимо выводить сумму в виде 12345,67 (дробные части копеек – отбросить).

После вывода графиков платежей вывести переплату по каждому виду из кредитов.

|  |  |  |
| --- | --- | --- |
| Входные данные | Вывод графика платежей | Переплата по типам кредита |
| amount = 300\_000d  rate = 16.9d  duration = 12 | 1: 25000,00 29225,00 27347,19  2: 25000,00 28872,92 27347,19  3: 25000,00 28520,83 27347,19  4: 25000,00 28168,75 27347,19  5: 25000,00 27816,67 27347,19  6: 25000,00 27464,58 27347,19  7: 25000,00 27112,50 27347,19  8: 25000,00 26760,42 27347,19  9: 25000,00 26408,33 27347,19  10: 25000,00 26056,25 27347,19  11: 25000,00 25704,17 27347,19  12: 25000,00 25352,08 27347,19 | 0,00 27462,50 28166,29 |