**Немного справочного материала.**

**5 способов скопировать массив в Java**

Итак, у вас есть два массива A и B и вам нужно скопировать элементы из A в B.

**Метод первый: ForLoop**

Здесь нам на помощь приходит старый-добрый цикл for:

**int**[] A = {1,2,4,4};

**int**[] B = **new** **int**[];

**for** (**int** i = 0; i < A.length; i++){

B[i] = A[i];

}

**Метод второй: .clone()**

Метод клонирования массива также может помочь добиться желаемого результата:

**int**[] A = {1,2,4,4};

**int**[] B = A.clone();//the clone method copies the content of A into B;

**Метод третий: System.arraycopy()**

Следующий способ — использование метода System.arraycopy(), который есть в пакете java.lang. Прежде чем мы перейдем к его применению, давайте обсудим его подпись:

**public** **static** **void** arraycopy(

Object src, //:source array, in this case A

**int** srcPos, //:the start index for copy, typically 0

Object dest, //:destination object in this case B.

**int** destPos, //:the index to place the copied elements

**int** length //:the length of the contents to be copied

);

Применение:

**int**[] A = {1,2,4,4};

**int**[] B = **new** **int**[];

System.arraycopy(A, 0, B, 0, A.length);

**Метод четвертый: Arrays.copyOf()**

Следующий вариант копирования, который мы обсудим, относится к классу Arrays из пакета java.utils. Обратите внимание на его подпись:

**public** **static** **int**[] copyOf(

**int**[] original, // :source array in this case A

**int** newLength // :the length of the contents to be copied

);

Применение:

**int**[] A = {1,2,4,4};

**int**[] B = Arrays.copyOf(A, 3);

**Метод пятый: Arrays.copyOfRange()**

Итак, это будет последний вариант, на который мы обратим внимание в этой публикации. Он тоже из класса Arrays, присутствующего в пакете java.utils. Еще раз посмотрим его подпись:

**public** **static** **int**[] copyOfRange​(

**int**[] original, // :source array in this case A

**int** from, //:the start index for copy, typically 0

**int** **to** // the end index exclusive

);

Применение:

**int**[] A = {1,2,3,4,5,6,7};

**int**[] B = Arrays.copyOfRange(A, 0, A.length);

**Задание.**

В классах можно добавлять методы и конструкторы по своему усмотрению.

Создайте класс Vector абстрактного вектора.

В этом классе если в методе участвуют два вектора, то ориентируемся на вектор меньшей размерности, обрезаем координаты вектора большей размерности.

В классе Vector следующие методы:

int dimension(); // размерность

double getComponent(int i); // возвращает компоненту вектора

void printVector(); выводит на консоль информацию о векторе

double scalar(Vector v);

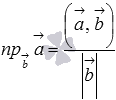
double len();

Vector multiply(double factor);

Vector add(Vector v);

Vector sub(Vector v);

double pr(Vector v); // возвращает проекция вектора на вектор v по формуле **, где - скалярное произведение векторов.**



Добавьте конструктор Vector (Vector v, int dim), который вектор v превратит в вектор заданной размерности dim.

Создать реализацию вектора в 2-мерном пространстве Vector2D на основе класса Vector. (Т.е. Vector2D наследует Vector.)

Создать конструктор Vector2D(double x, double y).

Добавить метод нахождения ортогонального вектора.

Добавить методы получения координат

double getX();

double getY().

Добавить метод нахождения векторной проекции вектора на другой вектор.

В классе Vector2D создать метод разложения по двум векторам. (Напоминание. Если вектор представлен в виде =х +у , где х и у — некоторые числа, то

говорят, что вектор разложен по векторам и . Числа х и у называются коэффициентами разложения.)

Но разложить вектор можно только по двум данным **неколлинеарным** векторам.

Значит если вектора коллинеарны, то метод должен «бросить» исключение.

Создадим в классе Vector2D метод

boolean pcollin(Vector2D v); // проверка на коллинеарность.

Создадим наш класс-исключение.

**public** **class** VcollinearException **extends** Exception {

**public** VcollinearException() {

**super**("Вектора коллинеарны");

}

}

Добавляем к методу разложения по двум векторам, что он кидает исключение

… **throws** VcollinearException

Внутри метода добавляем в начале.

Если вектора коллинеарны, то **throw** **new** VcollinearException();

В классе Vector2D переопределите метод equals() для сравнения значения полей.

С помощью класса Vector2D решить задачи.

1. Найти точку, симметричную данной точке относительной данной прямой.

Подсказка (может пригодиться).

Прямую зададим точкой P и вектором нормали . (**вектор нормали - это вектор, перпендикулярный искомой прямой.) S-точка, для которой будем искать симметричную точку относительно прямой.**

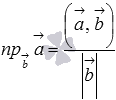
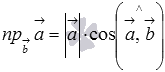
**Найдем вектор – векторная проекция вектора на вектор**  . От точки S отложим вектор  **и получим искомую точку.**

P

S

Симметричная точка

**Формула , позволяющая найти числовую проекцию вектора на прямую, направленную как вектор . Другой вид формулы , где - скалярное произведение векторов.**



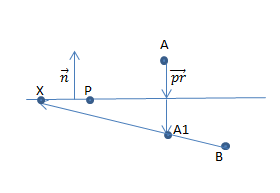
**2. Точки A и B лежат в разных полуплоскостях относительно прямой a. На прямой a найдите такую точку X, чтобы прямая a содержала биссектрису угла AXB.**

Подсказка (может пригодиться).

Пусть точка A1 – точка, симметричная A относительно прямой a. Тогда точка пересечения прямых a и A1B будет искомой.

Здесь по проекции находим длину самого вектора.

Надо сделать проверки: 1) лежат ли точки A и B в разных полуплоскостях (с помощью скалярного произведения); 2) прямые a и A1B непараллельны.



В отдельном классе Vector2DTest опишите метод main(...), в котором . найдем точку, симметричную данной, **точку X** ко второй задаче и разложим вектор по двум векторам, не забывая поймать исключение используется конструкция **try … catch**.