



Коллекции JAVA. Введение

Нужное



**У меня
есть план**



У меня есть план

1. Тип, который может быть чем угодно
2. Обобщения
3. Массивы и их проблемы
4. Самая «простая» коллекция
5. Функционал
6. Куда двигаться дальше



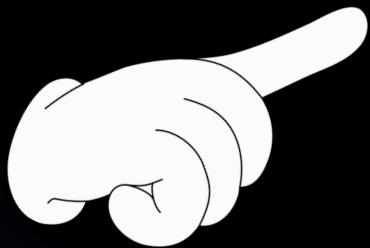


Object



Object

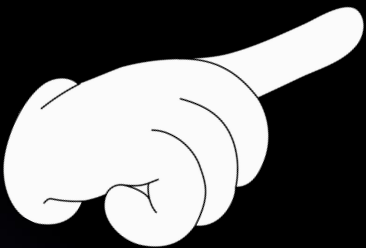
Тип данных **Object** – «всеу голова»



Object

Тип данных Object – «всему голова»

Упаковка – любой тип можно положить в переменную типа Object

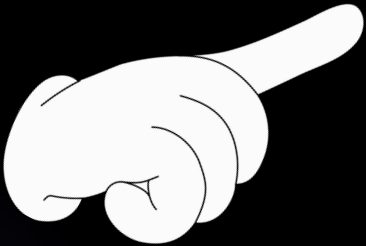


Object

Тип данных Object – «всему голова»

Упаковка – любой тип можно положить в переменную типа Object

Распаковка – преобразование Object-переменной в нужный тип



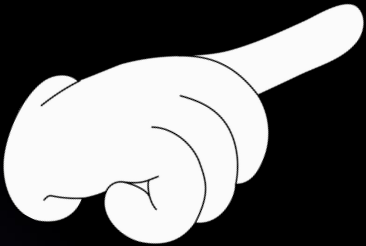
Object

Тип данных Object – «всему голова»

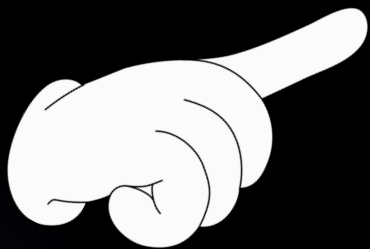
Упаковка – любой тип можно положить в переменную типа Object

Распаковка – преобразование Object-переменной в нужный тип

Иерархия типов – любой тип «ниже» Object'а



Object



Животные
↓
Млекопитающие
↓
Приматы
↓
Человек разумный



Object

Тип данных



Object

Тип данных

```
public class Ex01_object {  
    public static void main(String[] args) {  
        Object o = 1; GetType(o);    // java.lang.Integer  
        o = 1.2; GetType(o);        // java.lang.Double  
    }  
    static void GetType(Object obj) {  
        System.out.println(obj.getClass().getName());  
    }  
}
```



Object

```
public class Ex01_object {  
    public static void main(String[] args) {  
        System.out.println(Sum(1, 2));  
        System.out.println(Sum(1.0, 2));  
        System.out.println(Sum(1, 2.0));  
        System.out.println(Sum(1.2, 2.1));  
    }  
}
```

}



Object

```
public class Ex01_object {  
    public static void main(String[] args) {  
        System.out.println(Sum(1, 2));  
        System.out.println(Sum(1.0, 2));  
        System.out.println(Sum(1, 2.0));  
        System.out.println(Sum(1.2, 2.1));  
    }  
    static int Sum(int a, int b) { ...  
    }  
    static double Sum(double a, double b) { ...  
    }  
}
```

```
}
```



Object

```
public class Ex01_object {  
    public static void main(String[] args) {  
        System.out.println(Sum(1, 2));  
        System.out.println(Sum(1.0, 2));  
        System.out.println(Sum(1, 2.0));  
        System.out.println(Sum(1.2, 2.1));  
    }  
    static int Sum(int a, int b) { ...  
    }  
    static double Sum(double a, double b) { ...  
    }  
    static String Sum(String a, String b) { ...  
    }  
}
```

```
}
```



Object

```
public class Ex01_object {  
    public static void main(String[] args) {  
        System.out.println(Sum(1, 2));  
        System.out.println(Sum(1.0, 2));  
        System.out.println(Sum(1, 2.0));  
        System.out.println(Sum(1.2, 2.1));  
    }  
    static Object Sum(Object a, Object b) {  
  
        if (a instanceof Double && b instanceof Double) {  
            return (Object)((Double) a + (Double) b);  
        } else if (a instanceof Integer && b instanceof Integer) {  
            return (Object)((Integer) a + (Integer) b);  
        } else return 0;  
    }  
}
```



Object

```
public class Ex01_object {
    public static void main(String[] args) {
        System.out.println(Sum(1, 2));
        System.out.println(Sum(1.0, 2));
        System.out.println(Sum(1, 2.0));
        System.out.println(Sum("каша", "маша"));
    }
    static Object Sum(Object a, Object b) {

        if (a instanceof Double && b instanceof Double) {
            return (Object)((Double) a + (Double) b);
        } else if(a instanceof Integer && b instanceof Integer) {
            return (Object)((Integer) a + (Integer) b);
        } else return 0;
    }
}
```



Object

```
public class Ex01_object {  
    public static void main(String[] args) {  
        System.out.println(Sum(1, 2));  
        System.out.println(Sum(1.0, 2));  
        System.out.println(Sum(1, 2.0));  
        System.out.println(Sum(1.2, 2.1));  
    }  
}
```

```
}
```

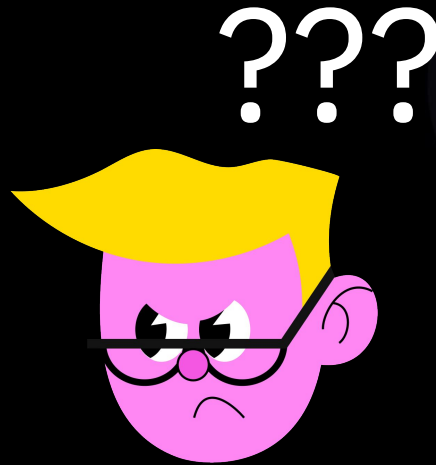


Object

Сотни пакетов и тысячи типов

Как проверять все?

Как описать все?



Object

```
public class Ex01_object {  
    public static void main(String[] args) {  
        System.out.println(Sum(1, 2));  
        System.out.println(Sum(1.0, 2));  
        System.out.println(Sum(1, 2.0));  
        System.out.println(Sum("каша", "маша"));  
    }  
    static Object Sum(Object a, Object b) {  
  
        if (a instanceof Double && b instanceof Double) {  
            return (Object)((Double) a + (Double) b);  
        } else if(a instanceof Integer && b instanceof Integer) {  
            return (Object)((Integer) a + (Integer) b);  
        } else return 0;  
    }  
}
```



Массивы



Массивы

Проблема. Как увеличить размер массива?



Массивы

Проблема. Как увеличить размер массива?

Есть массив из 2 элементов



Массивы

Демонстрация



Массивы

Проблема. Как увеличить размер массива?

Есть массив из 2 элементов

Например

```
public class Ex01_object {  
    public static void main(String[] args) {  
        int[] a = new int[] { 1, 9 };  
        int[] b = new int[3];  
        System.arraycopy(a, 0, b, 0, a.length);  
  
        for (int i : a) { System.out.printf("%d ", i);} // 1 9  
  
        for (int j : b) { System.out.printf("%d ", j);}  
        // 0 9 0 0 0 0 0 0 0 0  
    } }
```



Массивы

```
public class Ex01_object {
    static int[] AddItem(int[] array, int item) {
        int length = array.length;
        int[] temp = new int[length + 1];
        System.arraycopy(array, 0, temp, 0, length);
        temp[length] = item;
        return temp;
    }
    public static void main(String[] args) {
        int[] a = new int[] { 0, 9 };
        for (int i : a) { System.out.printf("%d ", i); }
        a = AddItem(a, 2);
        a = AddItem(a, 3);
        for (int j : a) { System.out.printf("%d ", j); }
    }
}
```



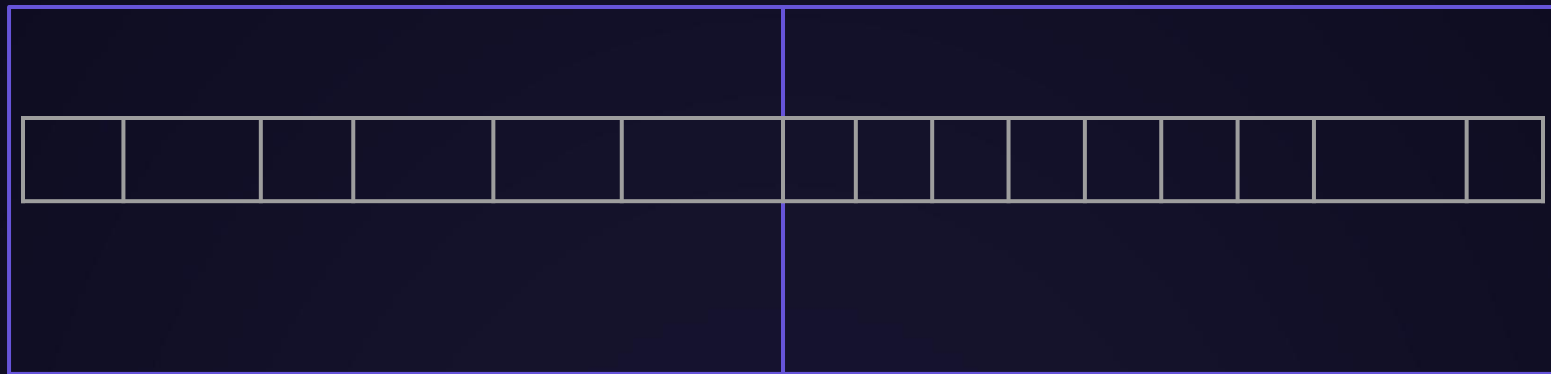
Массивы

RAM



Массивы

RAM



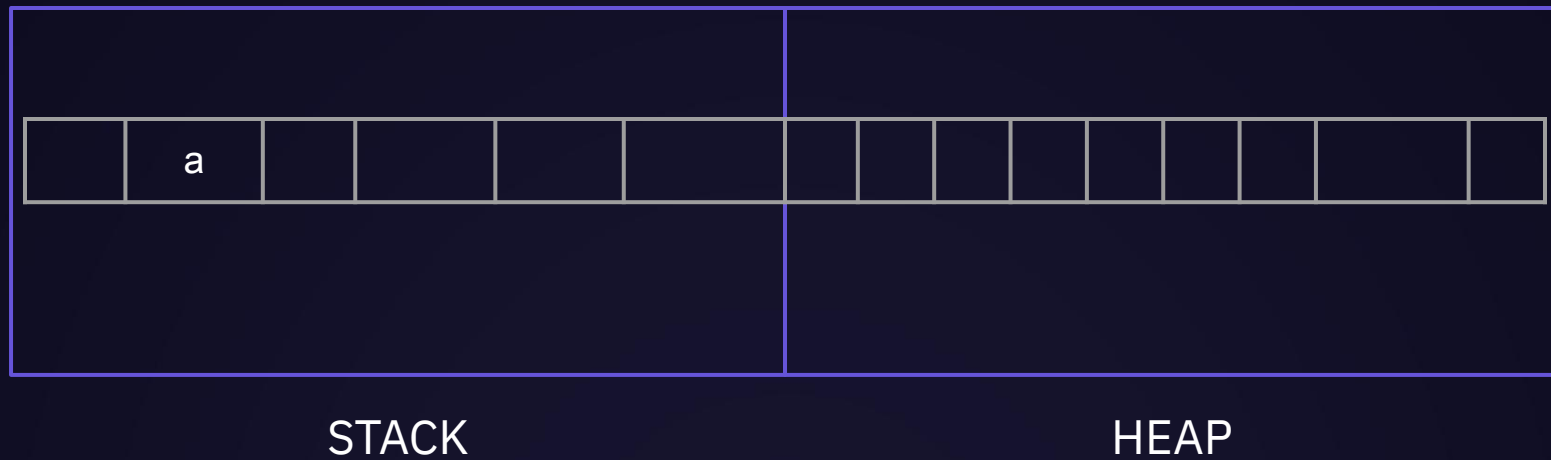
STACK

HEAP



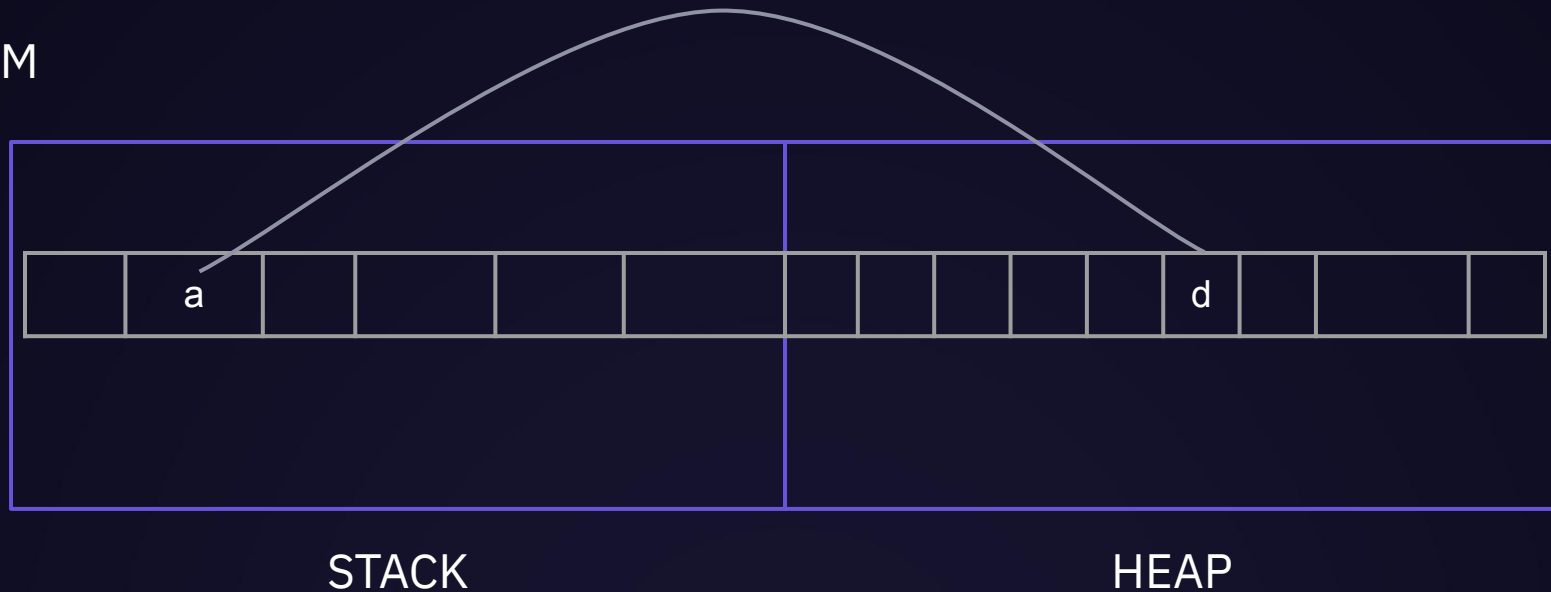
Массивы

RAM



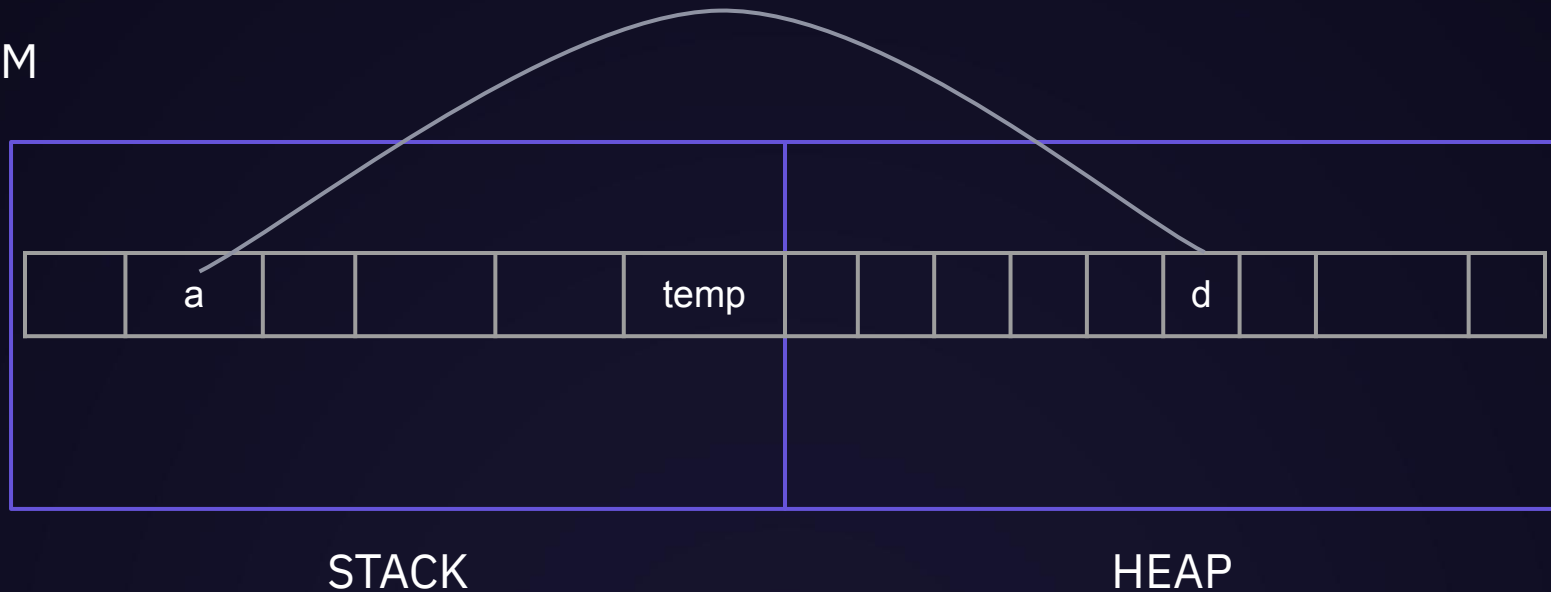
Массивы

RAM



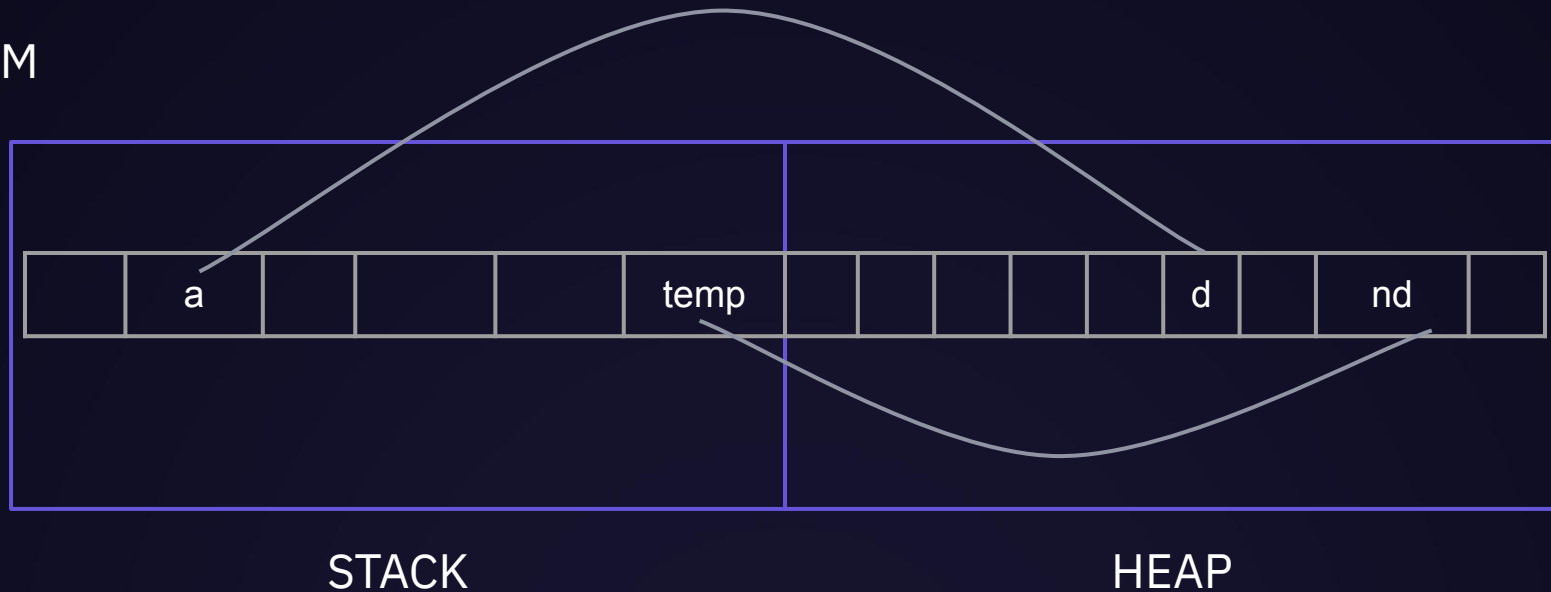
Массивы

RAM



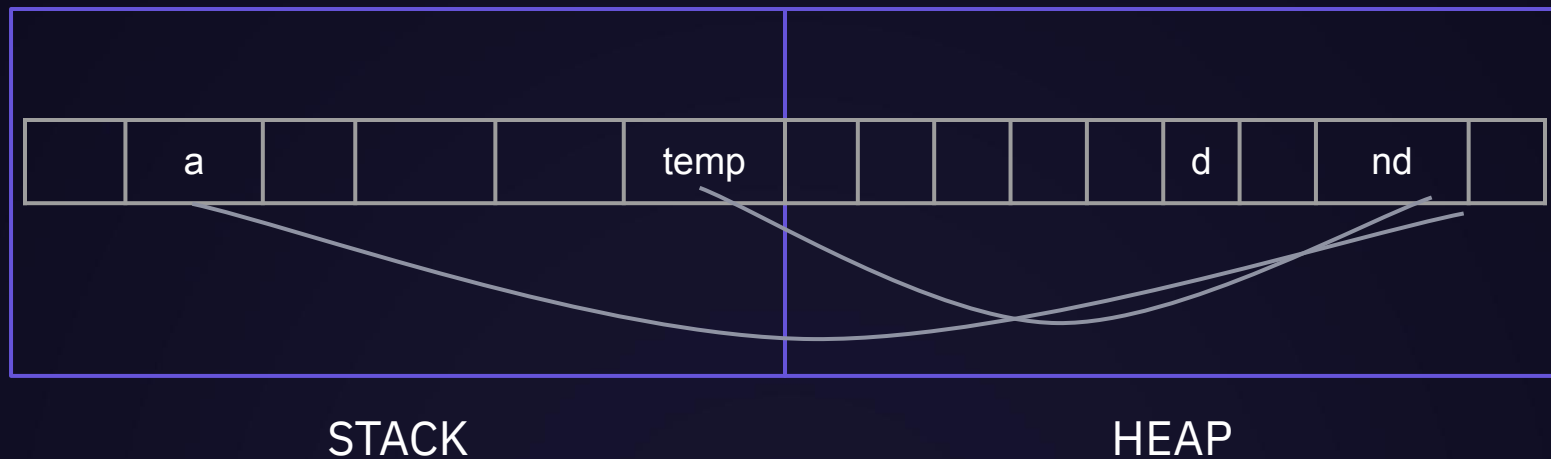
Массивы

RAM



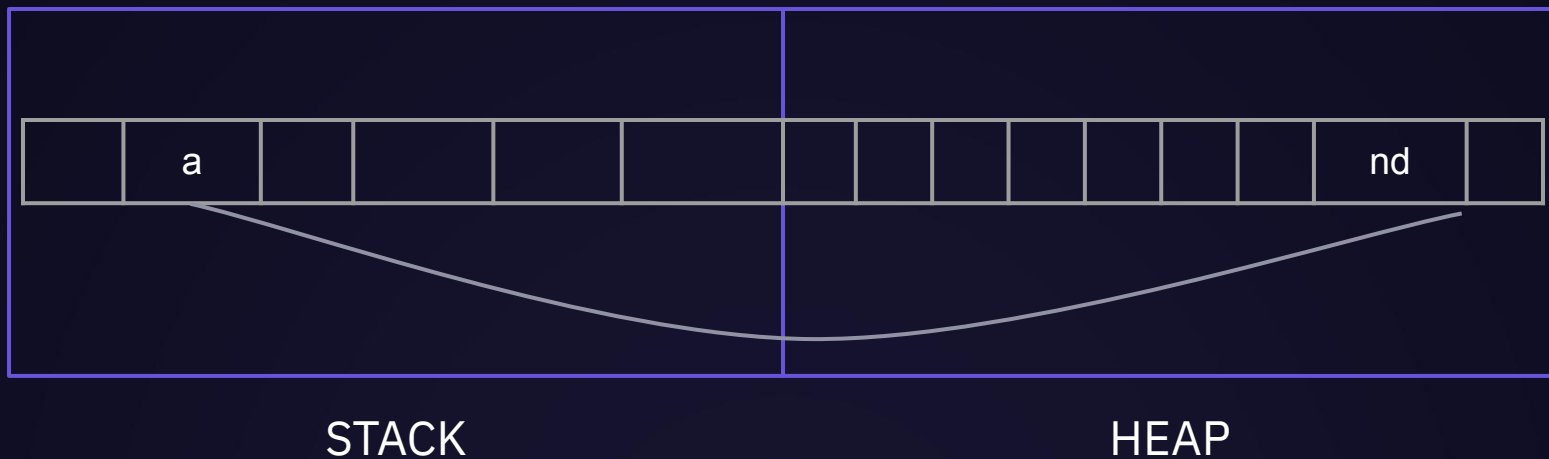
Массивы

RAM



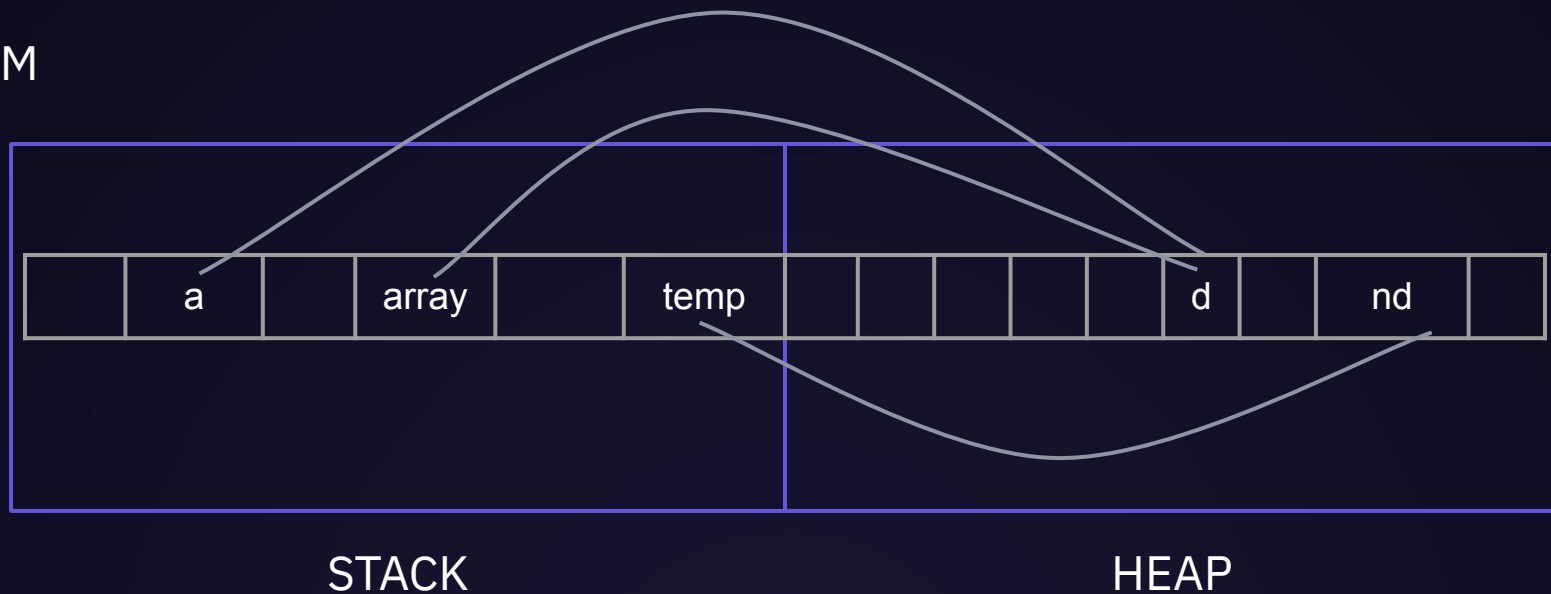
Массивы

RAM



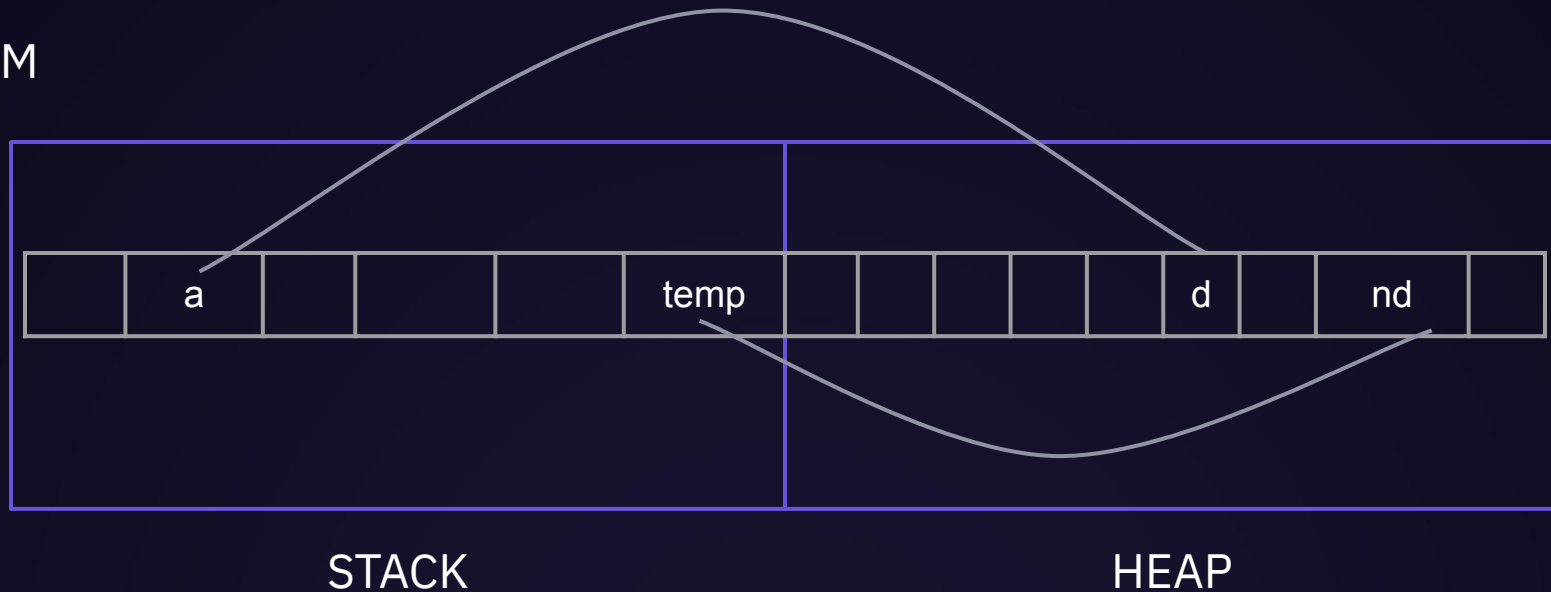
Массивы

RAM



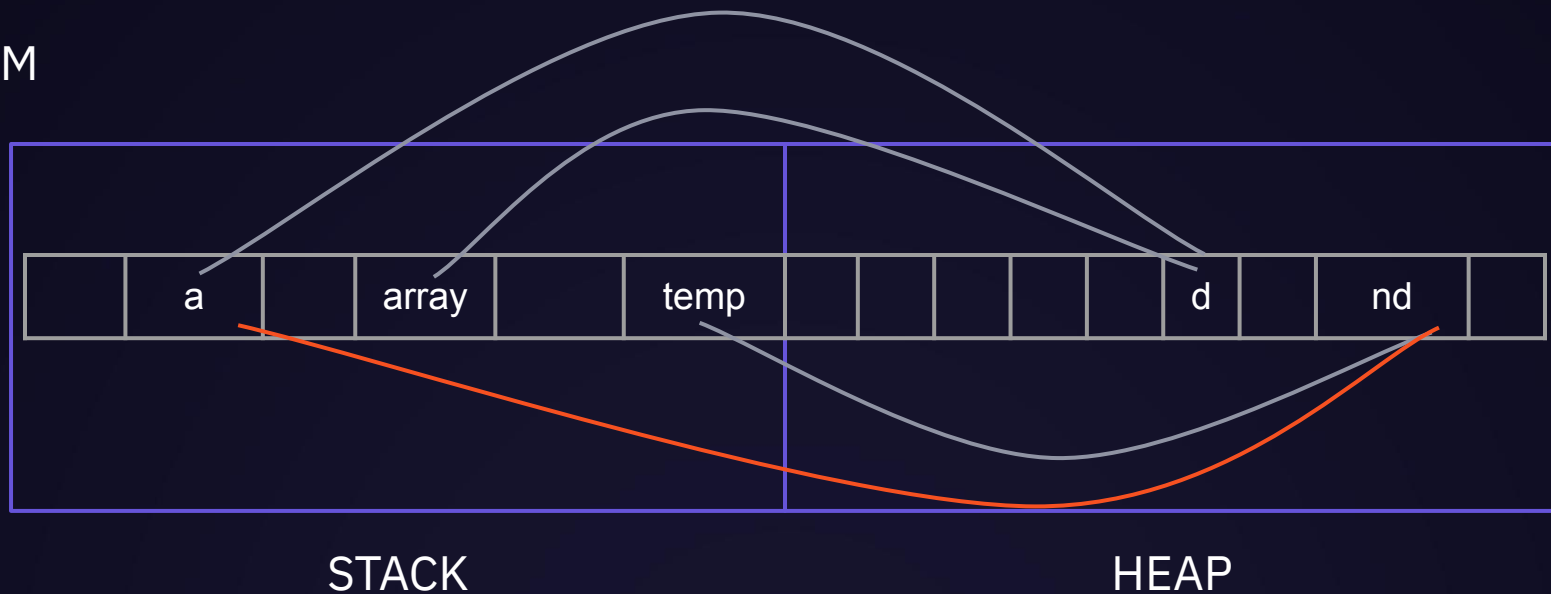
Массивы

RAM



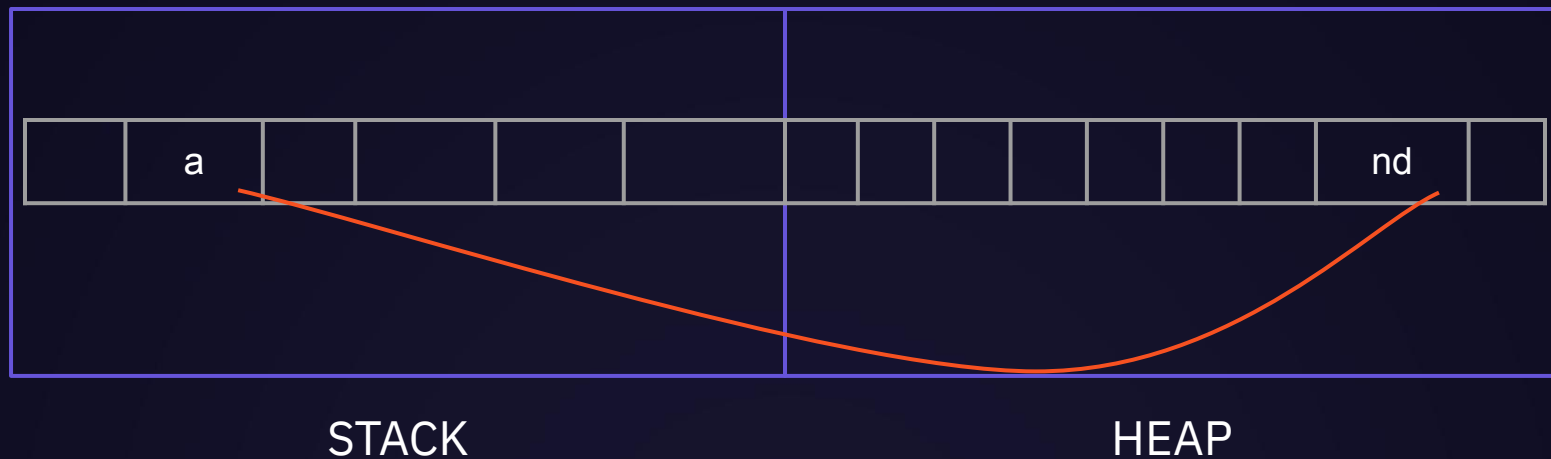
Массивы

RAM



Массивы

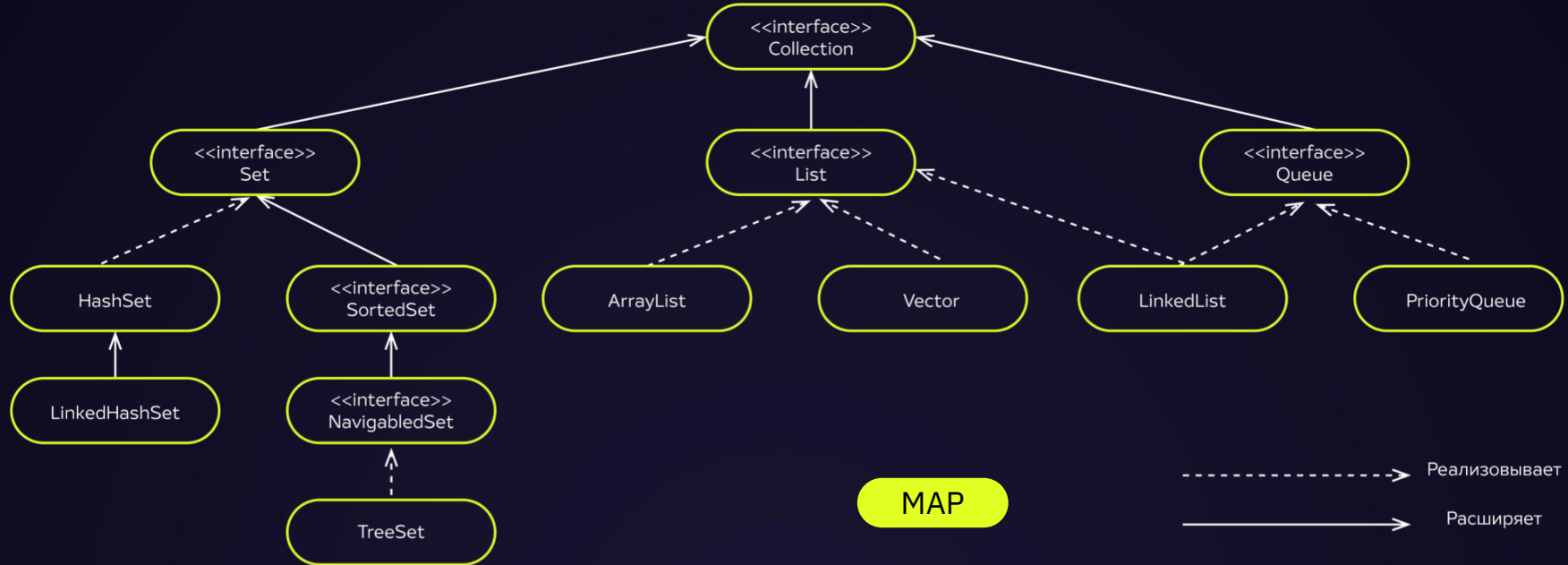
RAM



Коллекции



Иерархия коллекций



Иерархия коллекций. ArrayList

«Удобный массив»

Разные способы создания

```
ArrayList list = new ArrayList();
```



Массивы

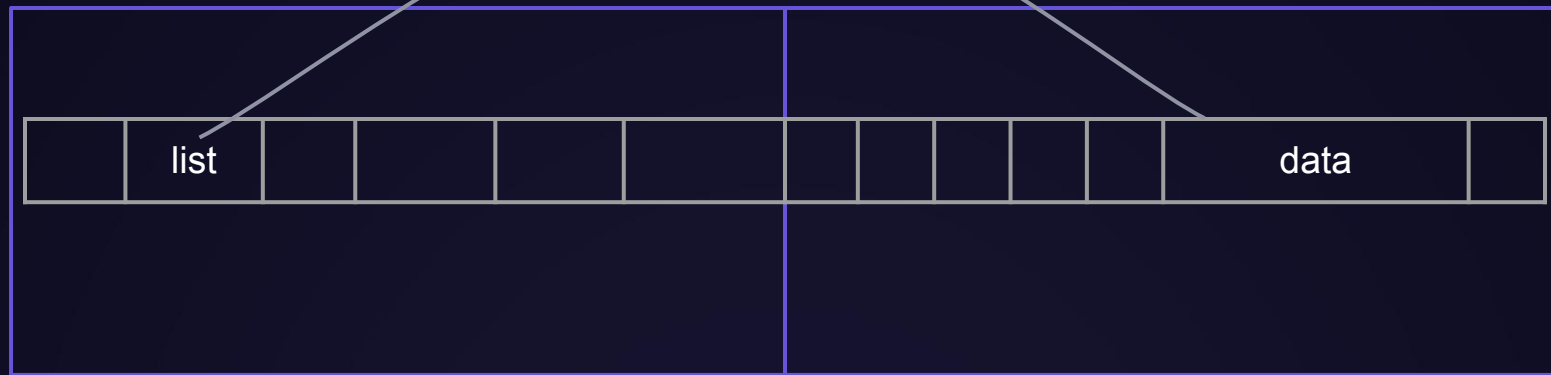
RAM



Массивы

RAM

NEW



STACK

HEAP





List



Иерархия коллекций. List

List – пронумерованный набор элементов.

Пользователь этого интерфейса имеет точный контроль над тем, где в списке вставляется каждый элемент.

Пользователь может обращаться к элементам по их целочисленному индексу (позиции в списке) и искать элементы в списке.

[URL](#)

ArrayList, LinkedList (Vector, Stack – устаревшие)



Иерархия коллекций. List

Демонстрация



Иерархия коллекций. ArrayList

«Удобный массив»

Разные способы создания

```
ArrayList<Integer> list1 = new ArrayList<Integer>();  
ArrayList<Integer> list2 = new ArrayList<>();  
ArrayList<Integer> list3 = new ArrayList<>(10);  
ArrayList<Integer> list4 = new ArrayList<>(list3);
```



Коллекции

```
import java.util.ArrayList;
import java.util.List;
public class Ex002_ArrayList {

    public static void main(String[] args) {
        List list = new ArrayList();
        list.add(2809);
        list.add("string line");
        for (Object o : list) {
            System.out.println(o);
            // Проблема извлечения данных
        }
    }
}
```



Коллекции. Row Type

```
import java.util.ArrayList;
import java.util.List;
public class Ex002_ArrayList {

    public static void main(String[] args) {
        List list = new ArrayList();
        list.add(2809);
        list.add("string line");
        for (Object o : list) {
            System.out.println(o);
            // Проблема извлечения данных
        }
    }
} // row type java
```



Коллекции. Save Type

```
import java.util.ArrayList;
import java.util.List;
public class Ex002_ArrayList {

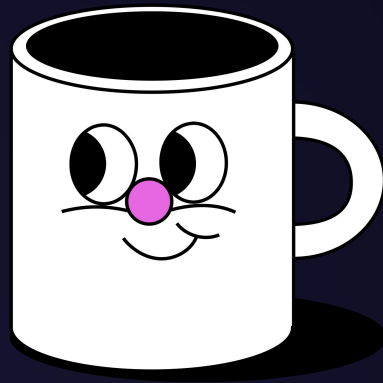
    public static void main(String[] args) {
        List<Integer> list = new ArrayList<Integer>();
        list.add(1);
        list.add(123);
        // list.add("string line");
        for (Object o : list) {
            System.out.println(o);
            // Проблема извлечения данных
        }
    }
}
```



Коллекции. Save Type

Ошибки на этапе компиляции лучше ошибок времени выполнения

Повторное использование кода



Иерархия коллекций. ArrayList

«Удобный массив»

Разные способы создания

```
ArrayList<Integer> list1 = new ArrayList<Integer>();  
ArrayList<Integer> list2 = new ArrayList<>();  
ArrayList<Integer> list3 = new ArrayList<>(10);  
ArrayList<Integer> list4 = new ArrayList<>(list3);
```



Коллекции. Функционал

add(args) – добавляет элемент в список (в т.ч. на нужную позицию)

get(pos) – возвращает элемент из списка по указанной позиции

indexOf(item) – первое вхождение или -1

lastIndexOf(item) – последнее вхождение или -1

remove(pos) – удаление элемента на указанной позиции и его возвращение

set(int pos, T item) – гivtoftn значение item элементу, который находится на позиции pos

void sort(Comparator) – сортирует набор данных по правилу

subList(int start, int end) – получение набора данных от позиции start до end



Коллекции. Функционал

Демонстрация



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex004 ArraysMethod {
    public static void main(String[] args) {
        int day = 29;
        int month = 9;
        int year = 1990;
        Integer[] date = { day, month, year };
        List<Integer> d = Arrays.asList(date);
        System.out.println(d); // [29, 9, 1990]
    }
}
```



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex004 ArraysMethod {
    public static void main(String[] args) {
        int day = 29;
        int month = 9;
        int year = 1990;
        Integer[] date = { day, month, year };
        List<Integer> d = Arrays.asList(date);
        System.out.println(d); // [29, 9, 1990]
    }
}
```

Демо by ref



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex005_ArraysMethod {
    public static void main(String[] args) {
        StringBuilder day = new StringBuilder("28");
        StringBuilder month = new StringBuilder("9");
        StringBuilder year = new StringBuilder("1990");
        StringBuilder[] date = { day, month, year };
        List<StringBuilder> d = Arrays.asList(date);
        System.out.println(d);
    }
}
```



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex005_ArraysMethod {
    public static void main(String[] args) {
        StringBuilder day = new StringBuilder("28");
        StringBuilder month = new StringBuilder("9");
        StringBuilder year = new StringBuilder("1990");
        StringBuilder[] date = { day, month, year };
        List<StringBuilder> d = Arrays.asList(date);
        System.out.println(d); // [29, 9, 1990]
    }
}
```



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex005_ArraysMethod {
    public static void main(String[] args) {
        StringBuilder day = new StringBuilder("28");
        StringBuilder month = new StringBuilder("9");
        StringBuilder year = new StringBuilder("1990");
        StringBuilder[] date = { day, month, year };
        List<StringBuilder> d = Arrays.asList(date);
        System.out.println(d); // [29, 9, 1990]
        month = new StringBuilder("09");
    }
}
```



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex005_ArraysMethod {
    public static void main(String[] args) {
        StringBuilder day = new StringBuilder("28");
        StringBuilder month = new StringBuilder("9");
        StringBuilder year = new StringBuilder("1990");
        StringBuilder[] date = { day, month, year };
        List<StringBuilder> d = Arrays.asList(date);
        System.out.println(d); // [29, 9, 1990]
        date[1] = new StringBuilder("09");
    }
}
```



Коллекции. Функционал

```
import java.util.Arrays;
import java.util.List;
public class Ex005_ArraysMethod {
    public static void main(String[] args) {
        StringBuilder day = new StringBuilder("28");
        StringBuilder month = new StringBuilder("9");
        StringBuilder year = new StringBuilder("1990");
        StringBuilder[] date = { day, month, year };
        List<StringBuilder> d = Arrays.asList(date);
        System.out.println(d); // [29, 9, 1990]
        date[1] = new StringBuilder("09");
        System.out.println(d); // [29, 09, 1990]
    }
}
```



Коллекции. Функционал

clear() – очистка списка

toString() – «конвертация» списка в строку

Arrays.asList – преобразует массив в список

containsAll(col) – проверяет включение всех элементов из col

removeAll(col) – удаляет элементы, имеющиеся в col

retainAll(col) – оставляет элементы, имеющиеся в col

toArray() – конвертация списка в массив Object'ов

toArray(type array) – конвертация списка в массив type

List.copyOf(col) – возвращает копию списка на основе имеющегося

List.of(item1, item2,...) – возвращает неизменяемый список



Коллекции. Функционал



Коллекции. Функционал

```
import java.util.List;

public class Ex006_ListOf {
    public static void main(String[] args) {
        Character value = null;
        List<Character> list1 =
            List.of('S', 'e', 'r', 'g', 'e', 'y');
        System.out.println(list1);
        // list1.remove(1); // java.lang.UnsupportedOperationException
        List<Character> list2 = List.copyOf(list1);
    }
}
```



Коллекции. Функционал

```
import java.util.List;

public class Ex006_ListOf {
    public static void main(String[] args) {
        Character value = null;
        List<Character> list1 =
            List.of('S', 'e', 'r', 'g', 'e', 'y');
        System.out.println(list1);
        // list1.remove(1); // java.lang.UnsupportedOperationException
        List<Character> list2 = List.copyOf(list1);
        // not null, immutable
    }
}
```



Итератор



Итератор

Получение итератора с целью более гибкой работы с данными [URL](#)

Интерфейс `Iterator<E>`. Итератор коллекцией. `Iterator` занимает место `Enumeration` в `Java Collections Framework`. Итераторы отличаются от перечислений двумя способами:

Итераторы позволяют вызывающей стороне удалять элементы из базовой коллекции во время итерации с четко определенной семантикой.

`hasNext()`, `next()`, `remove()`



Итератор

Получение итератора с целью более гибкой работы с данными [URL](#)

Интерфейс `Iterator<E>`. Итератор коллекцией. `Iterator` занимает место `Enumeration` в `Java Collections Framework`. Итераторы отличаются от перечислений двумя способами:

Итераторы позволяют вызывающей стороне удалять элементы из базовой коллекции во время итерации с четко определенной семантикой.

`hasNext()`, `next()`, `remove()`

`ListIterator<E>` [URL](#)

`hasPrevious()`, `E previous()`, `nextIndex()`, `previousIndex()`, `set(E e)`, `add(E e)`



Итератор

Демонстрация



Итератор

```
import java.util.*;
public class Ex007 Iterator {
    public static void main(String[] args) {
        List<Integer> list = List.of(1, 12, 123, 1234, 12345);

        for (int item : list) { System.out.println(item); }
        Iterator<Integer> col = list.iterator();

        while (col.hasNext()) {
            System.out.println(col.next());
        }
    }
}
```



Итератор

```
import java.util.*;
public class Ex007 Iterator {
    public static void main(String[] args) {
        List<Integer> list = List.of(1, 12, 123, 1234, 12345);

        for (int item : list) { System.out.println(item); }
        Iterator<Integer> col = list.iterator();


        while (col.hasNext()) {
            //System.out.println(col.next());
            col.next();
            col.remove();
        }
    }
}
```



ИТОГИ

Отрицание, гнев, торг...



Спасибо 
за внимание

