```java
public static void main(String[] args) {
    String s1 = "a1";
    String s2 = "a" + 1;
    System.out.println(s1.equals(s2));
}
```

```java
public static void main(String[] args) {
    int i;
    int o = 12;
    for (i = 1; i <= 10; i++) {
        o--;
    }
    System.out.println(o);
}
```

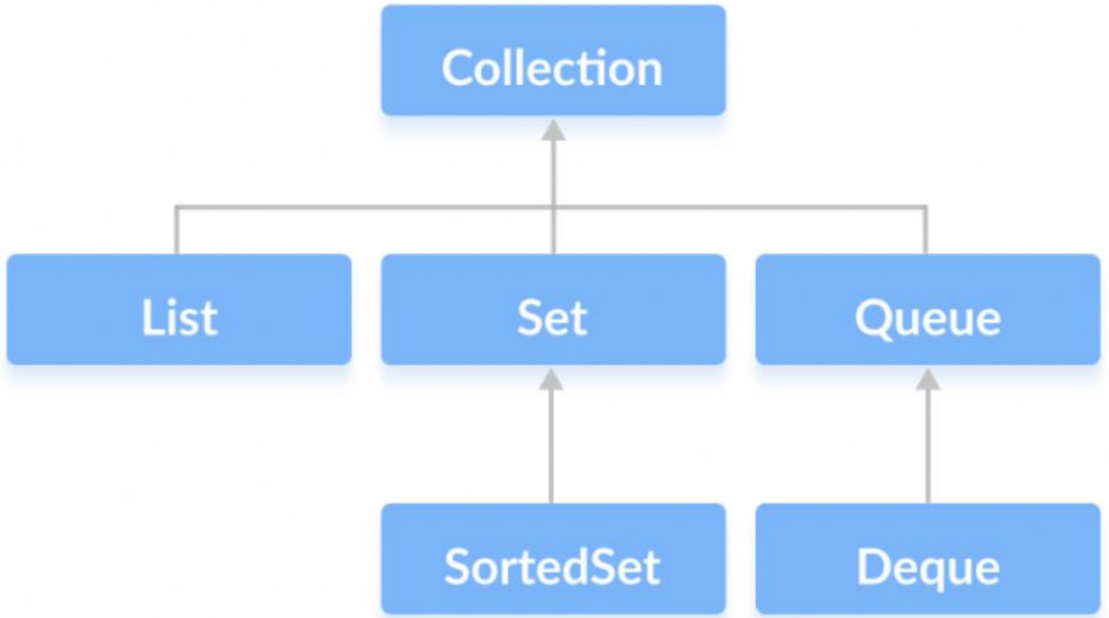
```java
public static void main(String[] args) {
    String s = "hello";
    s = s.concat("world");
    System.out.println(s);
}
```

```java
public static void main(String[] args) {
    String s = "Alex teacher";
    System.out.println(s.equals("alex teacher"));
}
```

```java
public static void main(String[] args) {
    int[][] arr = {{0, 1, 2}, {0, 1, 2, 3, 4, 5}};
    System.out.println(arr[1][2]);
}
```

```java
public static void main(String[] args) {
    String[] arr = {"a", "v", "u", "r", "b", "j"};
    String s = arr[5] + arr[0] + arr[1] + arr[0];
    System.out.println(s);
}
```

```java
public static void main(String[] args) {
    int[] list = {4, 5, 6, 7 ,8};
     System.out.println(list.length);
}
```


Collection
List
Set
Queue
SortedSet
Deque

метод

Arrays.asList()

формирует список на основе массива. Массив при этом используется для внутреннего представления списка. Таким образом сохраняется связь между списком и исходным массивом:

изменения в массиве отразятся на списке:

```
public static void main(String[] args) {
    String[] a = { "foo", "bar", "baz"};
    List<String> list = Arrays.asList(a);
    System.out.println(list); // [foo, bar, baz]

    a[0] = "aaa";
    System.out.println(list); // [aaa, bar, baz]
}
```

изменения в списке отразятся на массиве:

```
public static void main(String[] args) {
    String[] a = {"foo", "bar", "baz"};
    List<String> list = Arrays.asList(a);
    System.out.println(list); // [foo, bar, baz]

    list.set(0, "bbb");
    System.out.println(Arrays.toString(a)); // [bbb, bar, baz]
}
```

Collections.sort()

сортирует коллекцию

```
public static void main(String[] args) {
    List<String> list = Arrays.asList("красный", "синий", "зеленый");
    System.out.println("Перед сортировкой: " + list);
    Collections.sort(list);
    System.out.println("После сортировки: " + list);

    Collections.sort(list, Collections.reverseOrder());
    System.out.println("После обратной сортировки: " + list);
}
```

Методы Collections.reverse()

Переворачивает список в обратном порядке

```
public static void main(String[] args) {
    List<String> list = Arrays.asList("красный", "синий", "зеленый", "черный");
    System.out.println("Перед reversing: " + list);
    Collections.reverse(list);
    System.out.println("После reversing: " + list);
}
```

Методы Collections.shuffle()

Рандом

```
public static void main(String[] args) {
    List<String> list = Arrays.asList("красный", "синий", "зеленый", "черный");
    Collections.shuffle(list);
    System.out.println("После shuffling: " + list);
}
```

Методы Collections.max(), Collections.min()

Возвращает максимальное и минимальное значение

```
public static void main(String[] args) {
    List<Integer> list = Arrays.asList(2, 1, 5, 8, 9);
    System.out.println(Collections.max(list));
    System.out.println(Collections.min(list));
}
```

Методы Collections.copy()

Перезаписывает элементы одного листа, элементами другого листа

```
public static void main(String[] args) {
    List<Integer> src = Arrays.asList(1, 2, 3);
    List<Integer> dest = Arrays.asList(4, 5, 6, 7, 8);
    Collections.copy(dest, src);
    System.out.println(dest);
}
```

Методы Collections.frequency()

подсчитывает общее количество дублированных записей в листе

```
public static void main(String[] args) {
    Collection<String> collection = Arrays.asList("red", "cyan", "red");
    System.out.println(Collections.frequency(collection,  o: "red"));
}
```

```
public enum mathOperation {
    PLUS,
    MINUS,
    MULTIPLICATION,
    DIVISION
}
```

```java
static MathOperation getOperation() {
    Scanner sc = new Scanner(System.in);
    MathOperation operation;
    System.out.println("please write the operation +, -, * or /");

    String tmp = sc.nextLine();
    switch (tmp) {

        case "+":
            operation = MathOperation.PLUS;
            break;

        case "-":
            operation = MathOperation.MINUS;
            break;

        case "*":
            operation= MathOperation.MULTIPLICATION;
            break;

        case "/":
            operation= MathOperation.DIVISION;
            break;
        default:
            operation = MathOperation.PLUS;

    }
    return operation;
}
```

```java
static int result(int num1, int num2, MathOperation operation) {
    int res;

    switch (operation) {
        case PLUS:
            res = num1 + num2;
            break;
        case MINUS:
            res = num1 - num2;
            break;
        case MULTIPLICATION:
            res = num1 * num2;
            break;
        case DIVISION:
            res = num1 / num2;
            break;
        default:
            res = result(num1, num2, getOperation());
    }
    return res;
}
```

Maven - это инструмент для сборки Java проекта: скачивание дополнительных библиотек, компиляции, создания jar, генерации документации.

1. Зайдите на официальный сайт *Maven* в раздел загрузка и скачайте последнюю стабильную версию.
2. Распакуйте архив в инсталляционную директорию. Например в *"C:\Program Files\maven\"* в Windows или */opt/maven* в Linux.
3. Установите переменную окружения M2_HOME: в *"C:\Program Files\maven\"* .
4. Установите переменную окружения PATH: *"%M2_HOME%\bin"*.
5. Проверьте корректность установки, набрав в командной строке: mvn -version

Для MacOs в brew : brew install maven

1. Независимость от OS. Сборка проекта происходит в любой операционной системе. Файл проекта один и тот же.
2. Управление зависимостями. Редко какие проекты пишутся без использования сторонних библиотек (зависимостей). Эти сторонние библиотеки зачастую тоже в свою очередь используют библиотеки разных версий. Мавен позволяет управлять такими сложными зависимостями. Что позволяет разрешать конфликты версий и в случае необходимости легко переходить на новые версии библиотек.
3. Возможна сборка из командной строки. Такое часто необходимо для автоматической сборки проекта на сервере.
4. Хорошая интеграция со средами разработки.
5. Декларативное описание проекта (POM).
6. Огромный, поддерживаемый в актуальном состоянии репозиторий артефактов.
7. Модульная, расширяемая за счет плагинов архитектура, огромное количество плагинов

Недостатки Maven:

1. Сложность освоения.

2. Огромное количество плагинов (трудно сориентироваться).

3. Трудно разобраться если что то пошло не так (возникла ошибка).