String

java.lang 패키지

- 자바의 기본 제공 라이브러리(클래스 모음)

- import 생략 가능

- java.lang 패키지의 대표적인 클래스들
  - Object : 모든 자바 객체의 부모 클래스
  - String : 문자열
  - Integer, Long, Double : 래퍼 타입
  - Class : 클래스 메타 정보
  - System : 시스템과 관련된 기본 기능들을 제공

String 클래스 - 기본

- 자바에서 문자를 다루는 대표적인 타입: char, String

```
public class Main {
    public static void main(String[] args) {
        char[] charArr = new char[]{'j', 'a', 'v', 'a'};
        System.out.println(charArr);

        String str = "world";
        System.out.println("str = " + str);
    }
}
```

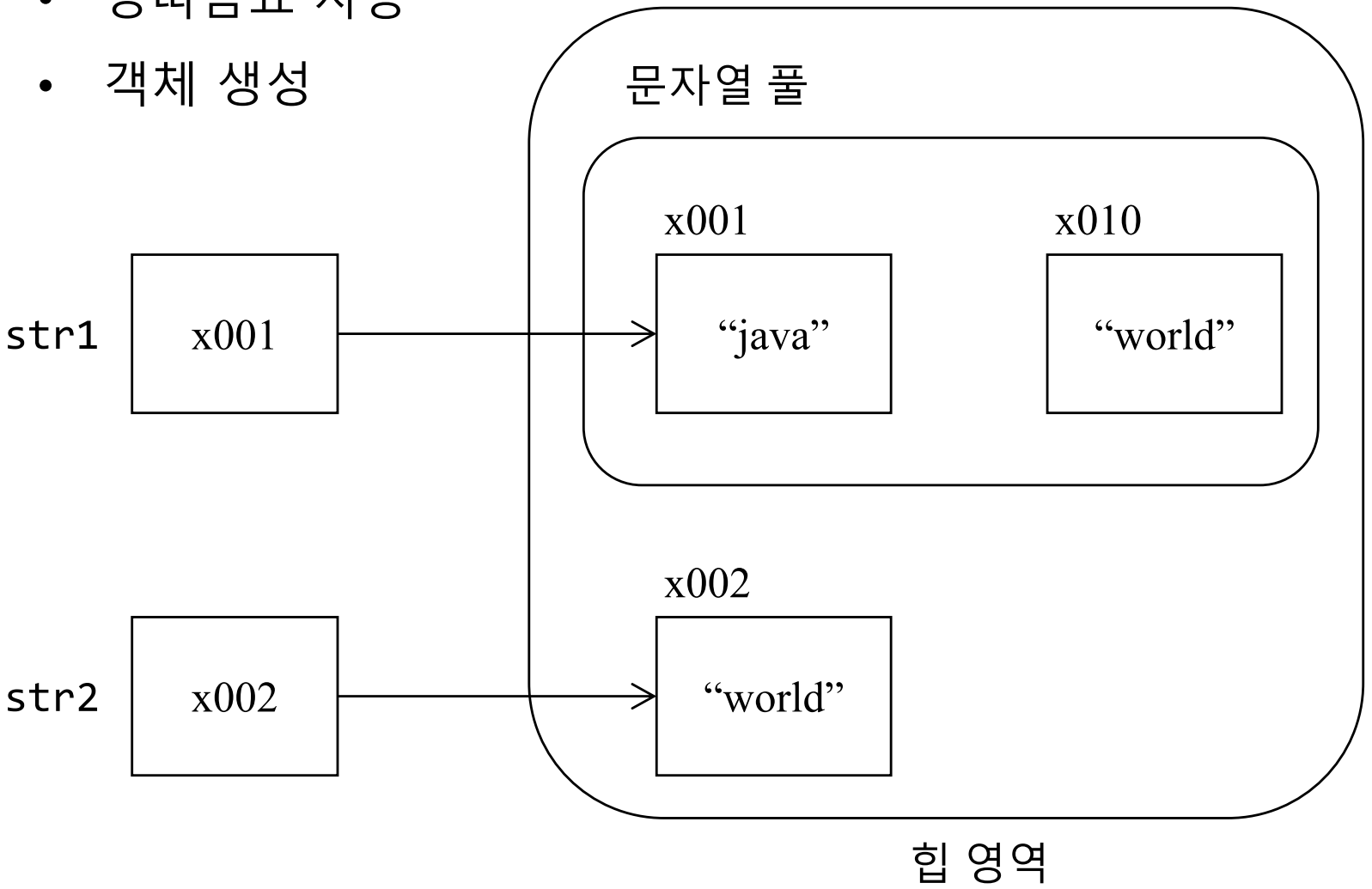
String 클래스 - 기본

- String 클래스를 통해 문자열을 생성하는 방법
  - 쌍따옴표 사용
  - 객체 생성

```
public class Main {
    public static void main(String[] args) {
        String str1 = "java";
        System.out.println("str1 = " + str1);
                                          참조형 변수의 + 연산
        String str2 = new String("world");
        System.out.println("str2 = " + str2);
    }
}
```

String 클래스 - 기본

- String 클래스를 통해 문자열을 생성하는 방법
  - 쌍따옴표 사용
  - 객체 생성



String 클래스 – 비교

- String 객체는 equals()를 이용하여 동등성 비교
  - **동일성(Identity)**: == 연산 사용. 두 객체의 참조 비교
  - **동등성(Equality)**: 두 객체가 논리적으로 같은지 확인

```
public class Main {
    public static void main(String[] args) {
        String str1 = "java";
        String str2 = new String("java");
        System.out.println(str1 == str2);

        String str3 = "java";
        System.out.println(str1 == str3);

        String str4 = new String("java");
        System.out.println(str2 == str4);
    }
}
```

```
/Users/qs/Library/Ja
false
true
false
```

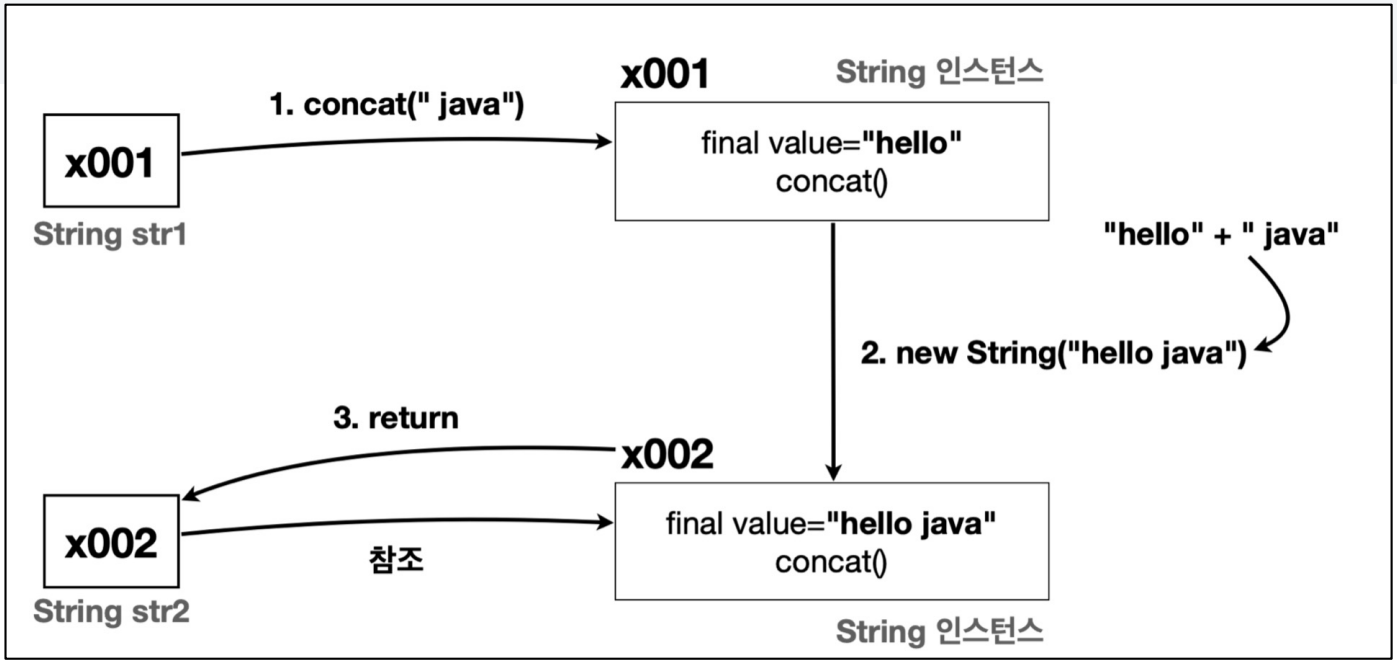
➔ String 객체의 비교는 equals()를 사용

String 클래스 – 불변 객체

- 생성 이후에 내부 문자열 값을 변경할 수 없음

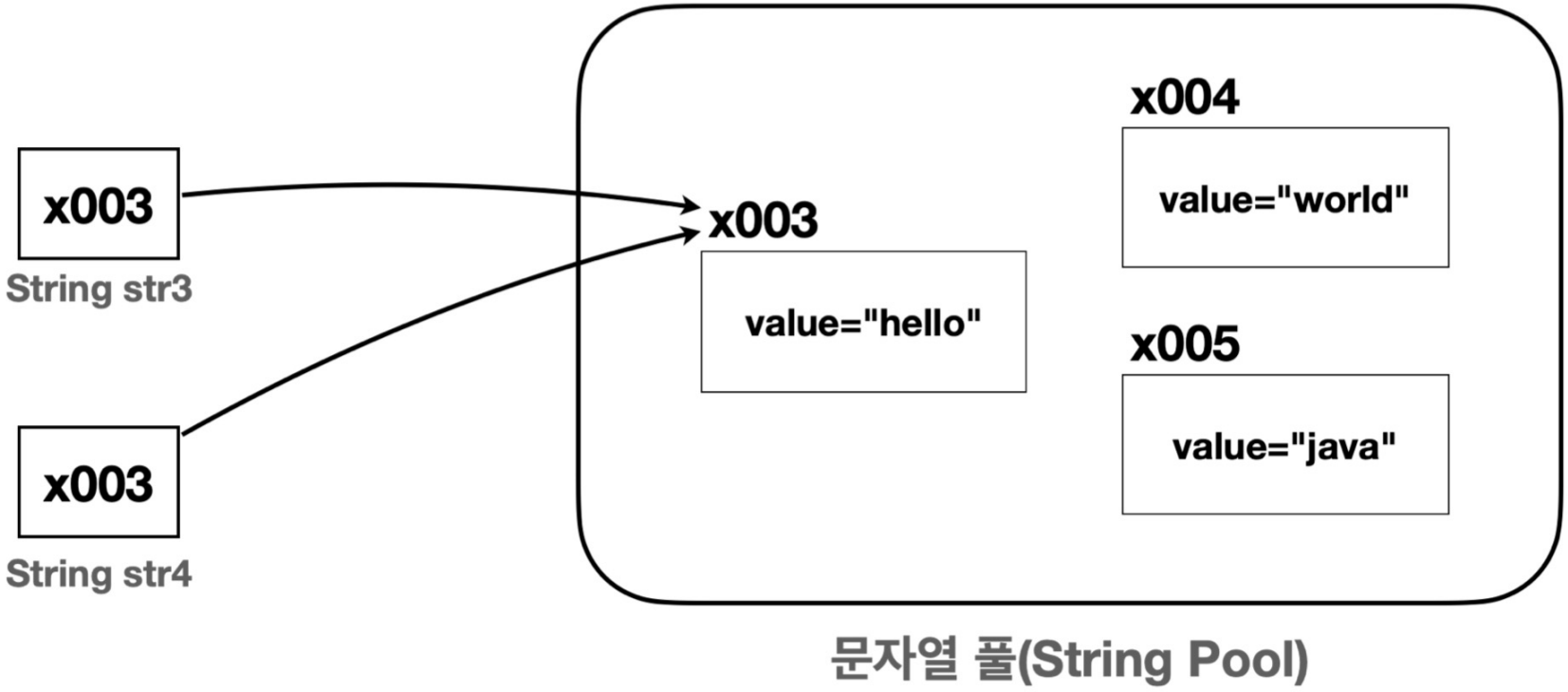
```java
public class StringImmutable2 {

    public static void main(String[] args) {
        String str1 = "hello";
        String str2 = str1.concat(" java");
        System.out.println("str1 = " + str1);
        System.out.println("str2 = " + str2);
    }
}
```



String이 불변으로 설계된 이유

- 기존에 문자열 풀에서 같은 문자를 참조하는 변수의 모든 문자가 함께 변경되어 버리는 문제가 발생



String 클래스 - 주요 메서드1

- 문자열 정보 조회
  - length() : 문자열의 길이를 반환한다.
  - isEmpty() : 문자열이 비어 있는지 확인한다. (길이가 0)
  - isBlank() : 문자열이 비어 있는지 확인한다. (길이가 0이거나 공백(Whitespace)만 있는 경우), 자바 11
  - charAt(int index) : 지정된 인덱스에 있는 문자를 반환한다.

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello, Java!";
        System.out.println("문자열의 길이: " + str.length());
        System.out.println("문자열이 비어 있는지: " + str.isEmpty());
        System.out.println("문자열이 비어 있거나 공백인지1: " + str.isBlank()); //Java11
        System.out.println("문자열이 비어 있거나 공백인지2: " + "    ".isEmpty());
        System.out.println("문자열이 비어 있거나 공백인지3: " + "    ". isBlank());

        char c = str.charAt(7);
        System.out.println("7번 인덱스의 문자: " + c);
    }
}
```

```
/Users/qs/Library/Java/JavaVirtualMachines
문자열의 길이: 12
문자열이 비어 있는지: false
문자열이 비어 있거나 공백인지1: false
문자열이 비어 있거나 공백인지2: false
문자열이 비어 있거나 공백인지3: true
7번 인덱스의 문자: J
```

String 클래스 - 주요 메서드1

- 문자열 비교

```java
public class Main {
    public static void main(String[] args) {
        String str1 = "Hello, Java!"; //대문자 일부 있음
        String str2 = "hello, java!"; //대문자 없음 모두 소문자
        String str3 = "Hello, World!";

        System.out.println("str1 equals str2: " + str1.equals(str2));
        System.out.println("str1 equalsIgnoreCase str2: " + str1.equalsIgnoreCase(str2));

        System.out.println("'b' compareTo 'a': " + "b".compareTo("a"));
        System.out.println("str1 compareTo str3: " + str1.compareTo(str3));
        System.out.println("str1 compareToIgnoreCase str2: " + str1.compareToIgnoreCase(str2));

        System.out.println("str1 starts with 'Hello': " + str1.startsWith("Hello"));
        System.out.println("str1 ends with 'Java!': " + str1.endsWith("Java!"));
    }
}
```

```
/Users/qs/Library/Java/JavaVirtualMachines
str1 equals str2: false
str1 equalsIgnoreCase str2: true
'b' compareTo 'a': 1
str1 compareTo str3: -13
str1 compareToIgnoreCase str2: 0
str1 starts with 'Hello': true
str1 ends with 'Java!': true
```

String 클래스 - 주요 메서드1

- 문자열 검색

```
public class Main {
    public static void main(String[] args) {
        String str = "Hello, Java! Welcome to Java world.";
        System.out.println("문자열에 'Java'가 포함되어 있는지: " + str.contains("Java"));
        System.out.println("'Java'의 첫 번째 인덱스: " + str.indexOf("Java"));
        System.out.println("인덱스 10부터 'Java'의 인덱스: " + str.indexOf("Java", 10));
        System.out.println("'Java'의 마지막 인덱스: " + str.lastIndexOf("Java"));
    }
}
```

```
문자열에 'Java'가 포함되어 있는지: true
'Java'의 첫 번째 인덱스: 7
인덱스 10부터 'Java'의 인덱스: 24
'Java'의 마지막 인덱스: 24
```

String 클래스 - 주요 메서드2

- 문자열 조작 및 변환

```java
public class Main {
    public static void main(String[] args) {
        String str = "Hello, Java! Welcome to Java";
        System.out.println("인덱스 7부터의 부분 문자열: " + str.substring(7));
        System.out.println("인덱스 7부터 12까지의 부분 문자열: " + str.substring(7, 12));
        System.out.println("문자열 결합: " + str.concat("!!!"));
        System.out.println("'Java'를 'World'로 대체: " + str.replace("Java", "World"));
        System.out.println("첫 번째 'Java'를 'World'으로 대체: " + str.replaceFirst("Java", "World"));
    }
}
```

```
/Users/qs/Library/Java/JavaVirtualMachines/openjdk-23.0.1/Contents/
인덱스 7부터의 부분 문자열: Java! Welcome to Java
인덱스 7부터 12까지의 부분 문자열: Java!
문자열 결합: Hello, Java! Welcome to Java!!!
'Java'를 'World'로 대체: Hello, World! Welcome to World
첫 번째 'Java'를 'World'으로 대체: Hello, World! Welcome to Java
```

String 클래스 - 주요 메서드2

- 문자열 조작 및 변환

```java
public class Main {
    public static void main(String[] args) {
        String strWithSpaces = " Java Programming ";
        System.out.println("소문자로 변환: " + strWithSpaces.toLowerCase());
        System.out.println("대문자로 변환: " + strWithSpaces.toUpperCase());
        System.out.println("공백 제거(trim): '" + strWithSpaces.trim() + "'");
        System.out.println("공백 제거(strip): '" + strWithSpaces.strip() + "'");
        System.out.println("앞 공백 제거(strip): '" + strWithSpaces.stripLeading() + "'");
        System.out.println("뒤 공백 제거(strip): '" + strWithSpaces.stripTrailing() + "'");
    }
}
```

```
/Users/qs/Library/Java/JavaVirtualMachines/o
소문자로 변환:  java programming
대문자로 변환:  JAVA PROGRAMMING
공백 제거(trim): 'Java Programming'
공백 제거(strip): 'Java Programming'
앞 공백 제거(strip): 'Java Programming '
뒤 공백 제거(strip): ' Java Programming'
```

String 클래스 - 주요 메서드2

- 문자열 분할 및 조합

```java
public class Main {
    public static void main(String[] args) {
        String str = "Apple,Banana,Orange";

        // split()
        String[] splitStr = str.split(",");
        for (String s : splitStr) {
            System.out.println(s);
        }

        // join()
        String joinedStr = String.join("-", "A", "B", "C");
        System.out.println("연결된 문자열: " + joinedStr);

        // 문자열 배열 연결
        String result = String.join("-", splitStr);
        System.out.println("result = " + result);
    }
}
```

```
/Users/qs/Library/Java/JavaVirtua
Apple
Banana
Orange
연결된 문자열: A-B-C
result = Apple-Banana-Orange
```

String 클래스 - 주요 메서드2

- 기타 유틸리티

```
숫자의 문자열 값: 100
불리언의 문자열 값: true
객체의 문자열 값: java.lang.Object@5f184fc6
빈문자열 + num:100
문자열을 문자 배열로 변환: [C@3feba861
Hello, Java!
```

```java
public class Main {
    public static void main(String[] args) {
        int num = 100;
        boolean bool = true;
        Object obj = new Object();
        String str = "Hello, Java!";

        // valueOf 메서드
        String numString = String.valueOf(num);
        System.out.println("숫자의 문자열 값: " + numString);
        String boolString = String.valueOf(bool);
        System.out.println("불리언의 문자열 값: " + boolString);
        String objString = String.valueOf(obj);
        System.out.println("객체의 문자열 값: " + objString);
        // 다음과 같이 간단히 변환할 수 있음 (문자 + x -> 문자x)
        String numString2 = "" + num; System.out.println("빈문자열 + num:" + numString2);

        // toCharArray 메서드
        char[] strCharArray = str.toCharArray();
        System.out.println("문자열을 문자 배열로 변환: " + strCharArray);
        for (char c : strCharArray) {
            System.out.print(c);
        }
        System.out.println();
    }
}
```

String 클래스 - 주요 메서드2

- 기타 유틸리티

```java
public class Main {
    public static void main(String[] args) {
        int num = 100;
        boolean bool = true;
        String str = "Hello, Java!";

        //format 메서드
        String format1 = String.format("num: %d, bool: %b, str: %s", num, bool, str);
        System.out.println(format1);

        String format2 = String.format("숫자: %.2f", 10.1234);
        System.out.println(format2);

        // printf
        System.out.printf("숫자: %.2f\n", 10.1234);

        // matches 메서드
        String regex = "Hello, (Java!|World!)";
        System.out.println("'str'이 패턴과 일치하는가? " + str.matches(regex));
    }
}
```

```
num: 100, bool: true, str: Hello, Java!
숫자: 10.12
숫자: 10.12
'str'이 패턴과 일치하는가? true
```

불변인 String 클래스의 단점

- 문자를 더하거나 변경할 때 마다 새로운 객체를 생성
- 위의 상황에서 많은 String 객체를 만들고, GC ➔ 컴퓨터의 CPU, 메모리를 자원을 더 많이 사용

```
"A" + "B"
String("A") + String("B") //문자는 String 타입이다.
String("A").concat(String("B"))//문자의 더하기는 concat을 사용한다.
new String("AB") //String은 불변이다. 따라서 새로운 객체가 생성된다.
```

StringBuilder – 가변 String

- 해결책: 가변 String = **StringBuilder**
  - StringBuilder 는 보통 문자열을 변경하는 동안만 사용하다가 문자열 변경이 끝나면 안전한(불변) String 으로 변환하는 것이 좋다.

```
public class Main {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        sb.append("A");
        sb.append("B");
        sb.append("C");
        sb.append("D");
        System.out.println("sb = " + sb);

        sb.insert(4, "Java");
        System.out.println("insert = " + sb);
        sb.delete(4, 8);
        System.out.println("delete = " + sb);

        sb.reverse();
        System.out.println("reverse = " + sb);

        //StringBuilder -> String
        String string = sb.toString();
        System.out.println("string = " + string);
    }
}
```

```
/Users/qs/Library/Java/J
sb = ABCD
insert = ABCDJava
delete = ABCD
reverse = DCBA
string = DCBA
```

자바의 String 최적화

문자열 리터럴 최적화

컴파일 전

```
String helloWorld = "Hello, " + "World!";
```

컴파일 후

```
String helloWorld = "Hello, World!";
```

String 변수 최적화

문자열 변수의 경우 그 안에 어떤 값이 들어있는지 컴파일 시점에는 알 수 없기 때문에 단순하게 합칠 수 없다.

```
String result = str1 + str2;
```

이런 경우 예를 들면 다음과 같이 최적화를 수행한다. (최적화 방식은 자바 버전에 따라 달라진다.)

```
String result = new StringBuilder().append(str1).append(str2).toString();
```

String 최적화가 어려운 경우

```
public class Main {
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();

        String result = "";
        for (int i = 0; i < 100000; i++) {
            result += "Hello Java ";
        }
        long endTime = System.currentTimeMillis();

        System.out.println("result = " + result);
        System.out.println("time = " + (endTime - startTime) + "ms");

    }
}
```

```
Hello Java Hello Java He
time = 2800ms
```

```
public class Main {
    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 100000; i++) {
            sb.append("Hello Java ");
        }
        String result = sb.toString();
        long endTime = System.currentTimeMillis();
        System.out.println("result = " + result);
        System.out.println("time = " + (endTime - startTime) + "ms");
    }
}
```

```
Hello Java Hello Java
time = 3ms
```