Strings

Strings

- Sequence of characters and not a primitive data type
- **Example:** Hello
- In java, strings are objects.
- The class **“String”** comes under java.lang package.
- Internally string is implemented as a character array.
- **E.g:**

H	e	l	l	0

How to create a string?

Two ways to create a string:

1) String literal

2) Using new keyword

String literal:

String str = “Hello”;

Using new Keyword:

String s = new String(“Hello”);

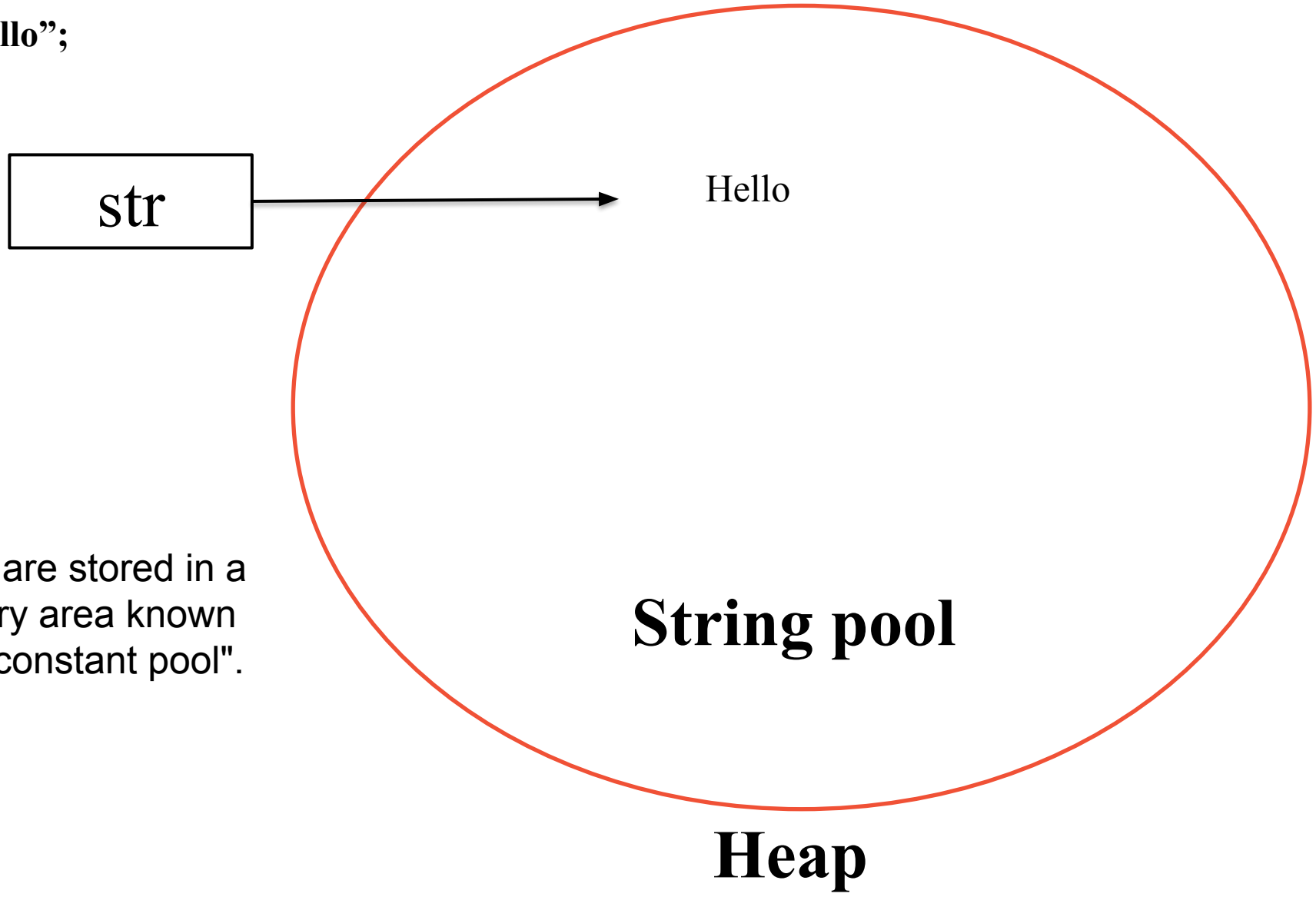
Basic string Program

```
// Predict the output

class Main{
    public static void main(String args[]){
        String str = "Programming";
        String s = new String("World");
        System.out.println(str);
        System.out.println(s);
    }
}
```

Memory allocation - String Literal

String str = “Hello”;

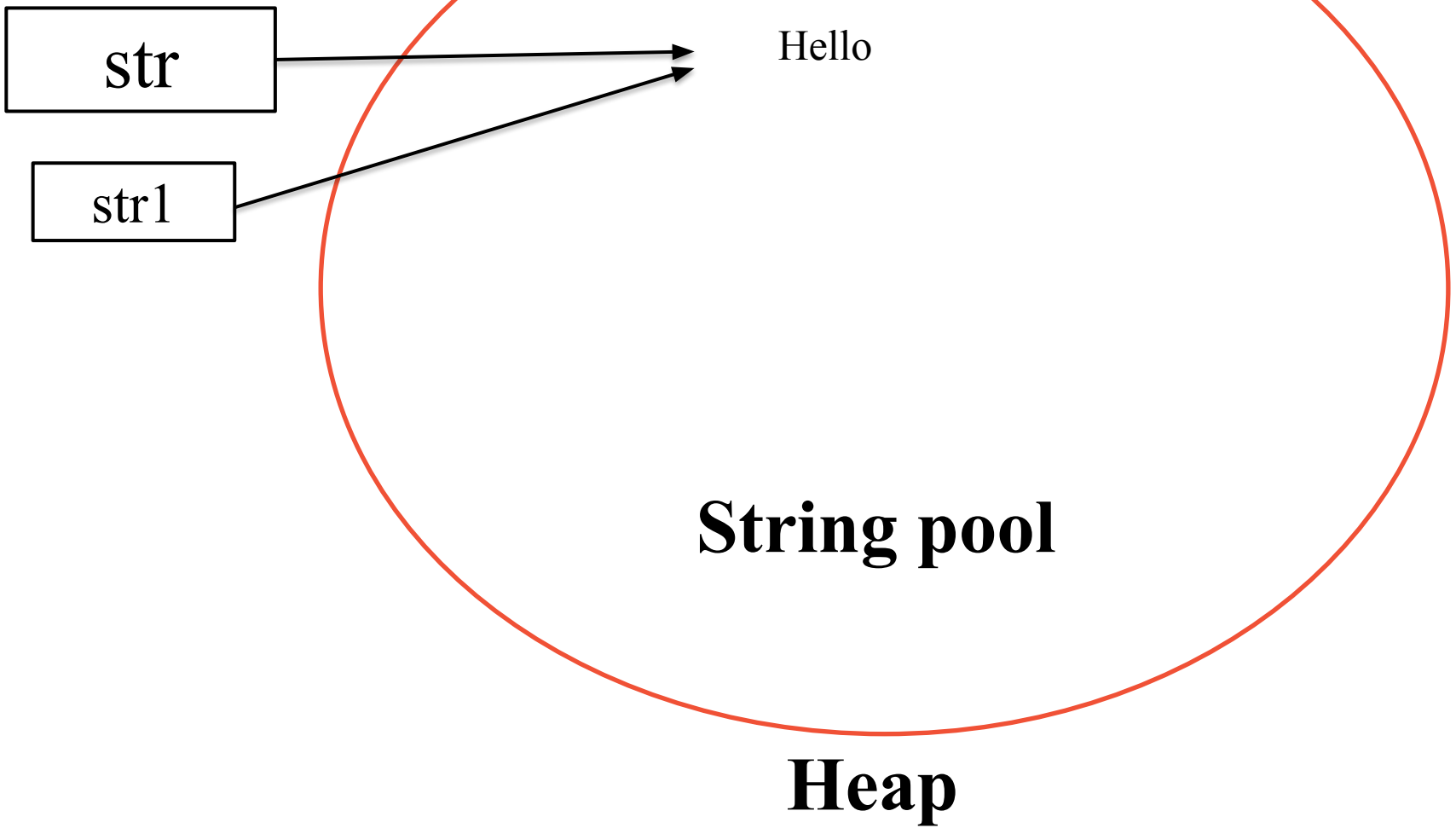


String objects are stored in a special memory area known as the "string constant pool".

Memory allocation - String Literal(Contd…

String str1 = “Hello”;

How it is working in the backend?



Memory allocation - String Literal(Explanation)

If we create an object using String literal. It may return an existing object from the String pool, if it already exists.
Otherwise, it will create a new String object and put in the string pool for future re-use.

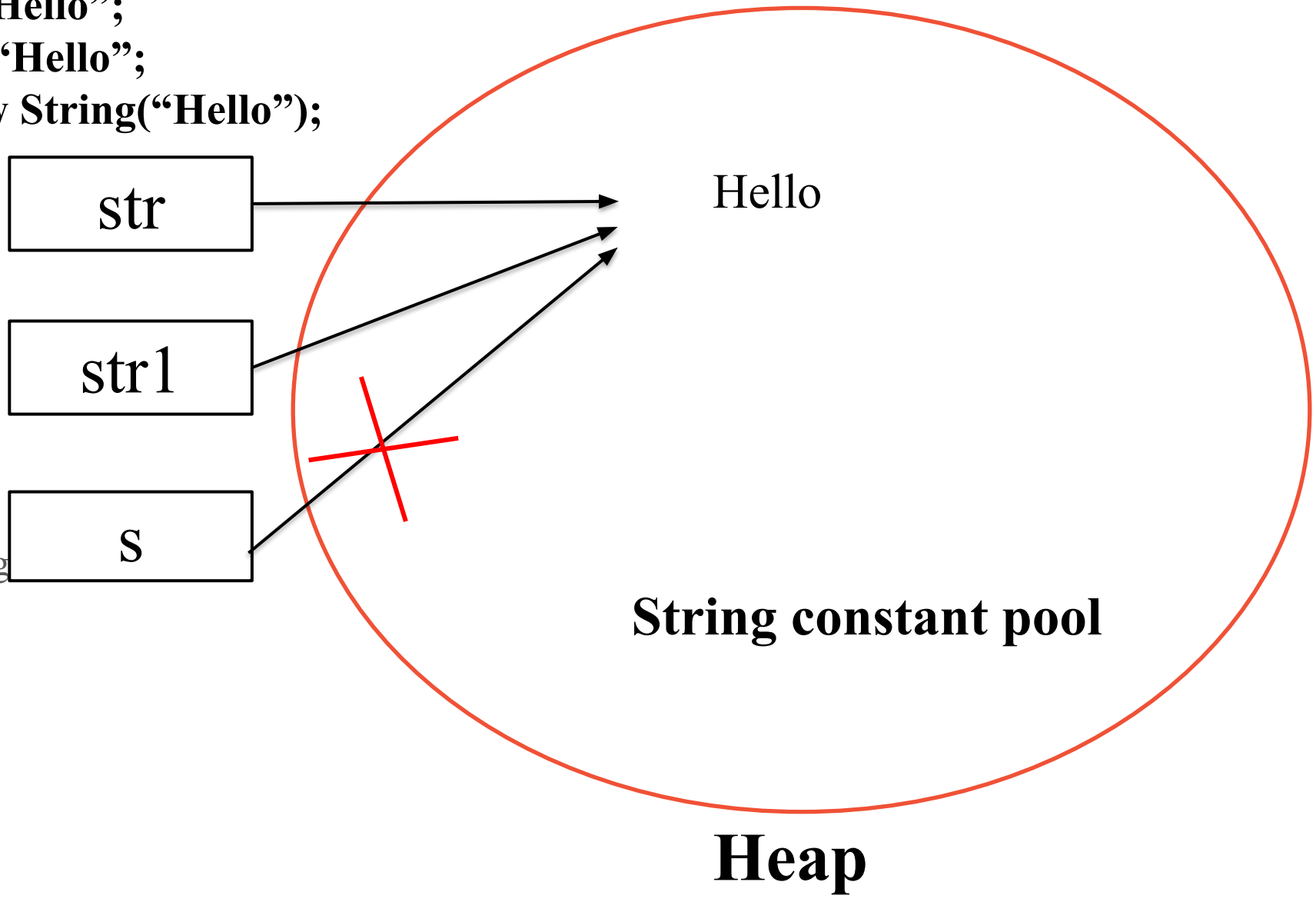
The same reference will be assigned to str and str1.

Why Java uses the concept of String literal?

To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

Memory allocation – Using new Keyword

String str = “Hello”;
String str1 = “Hello”;
String s = new String(“Hello”);



How it is working

Memory allocation – Using new Keyword(Explanation)

new - String s = new String(“Hello”)

In such case, JVM will create a new string object in heap memory, and the literal “Hello" will be placed outside the string constant pool.

The variable “s” will refer to the object in a heap.

Therefore, if we compare performance of string literal and string object, string object will always take more time to execute than string literal because it will construct a new string every time it is executed.

Note: Execution time is compiler dependent.

Predict the output

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

```
true
false
```

Predict the output

```
// Predict the output of the below code

class Main{
    public static void main(String args[]){
        String str1 = "Hello";
        String str2 = "Hello";
        String s1 = new String("Hello");
        String s2 = new String("Hello");
        System.out.println(str1 == str2);
        System.out.println(str2 == s1);
        System.out.println(s1 == s2);
    }
}
```

```
true
false
false
```

Predict the output

```
// Predict the output of the below code

class Main{
    public static void main(String args[]){
        String s1 = new String("C2TC");
        String s2 = new String("C2TC");
        System.out.println(s1 == s2);
        System.out.println(s1.equals(s2));
    }
}
```

```
false
true
```