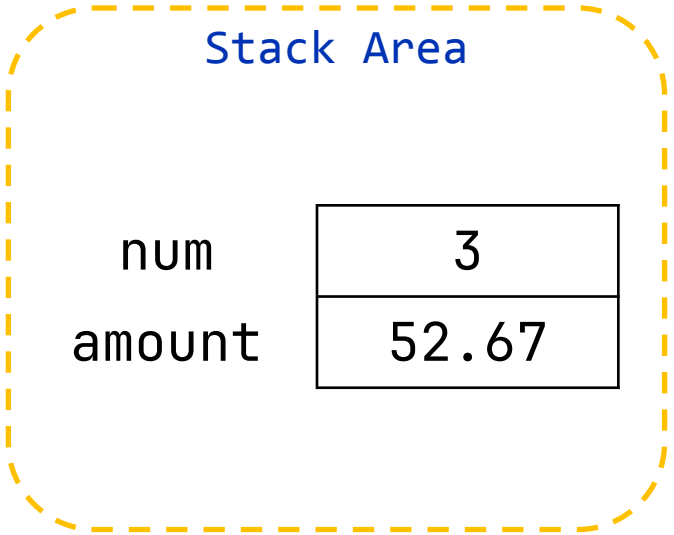
03_1 Reference Types

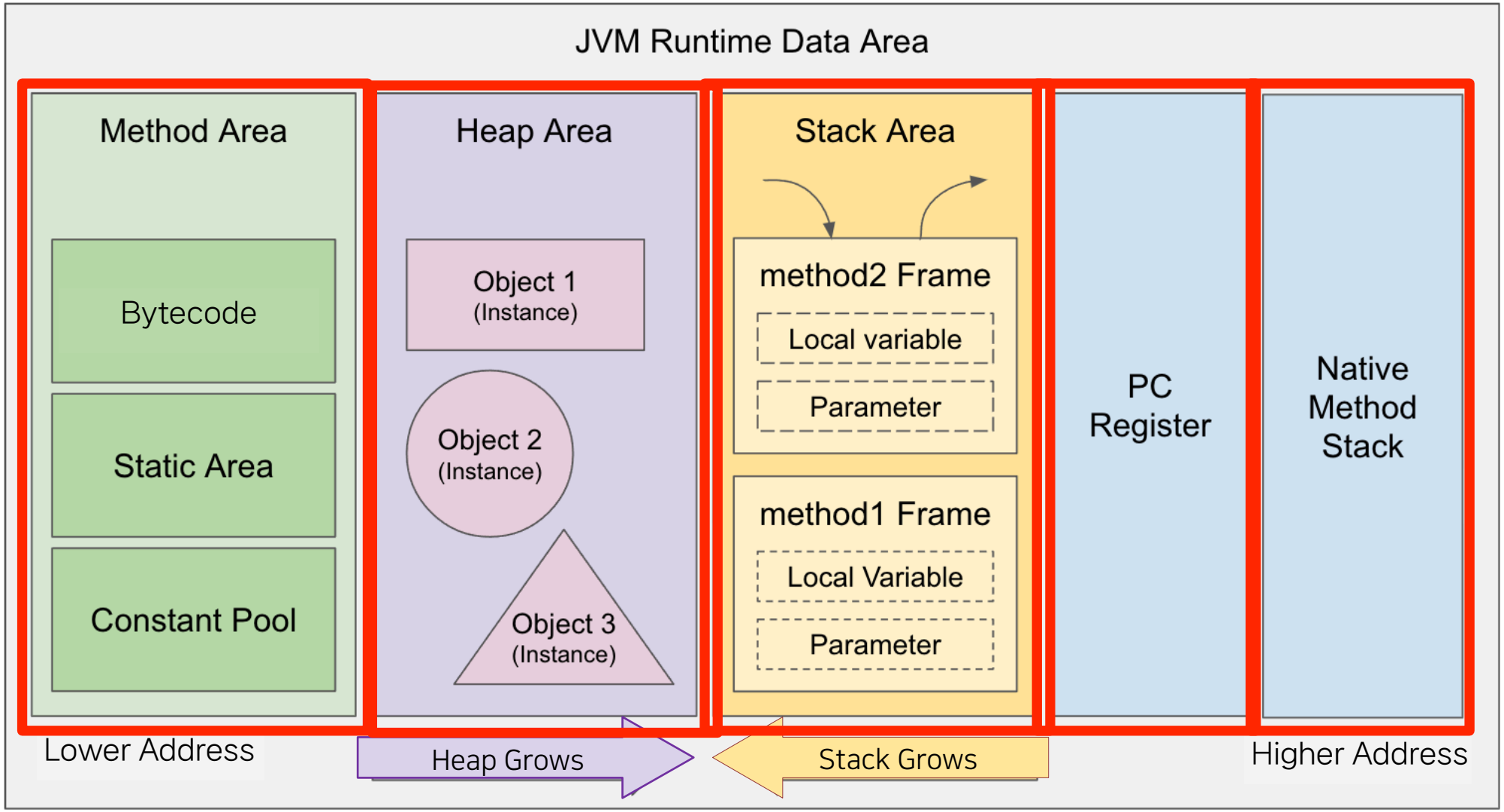
Object-Oriented Programming

Primitive Types

- Primitive Types
  - Integer Types
    - byte, char, short, int, long
  - Real Number Types
    - float, double
  - Caracter Type
    - char
  - Logical Types
    - boolean
- Ex)
  - `int num = 3;`
  - `float amount = 52.67;`

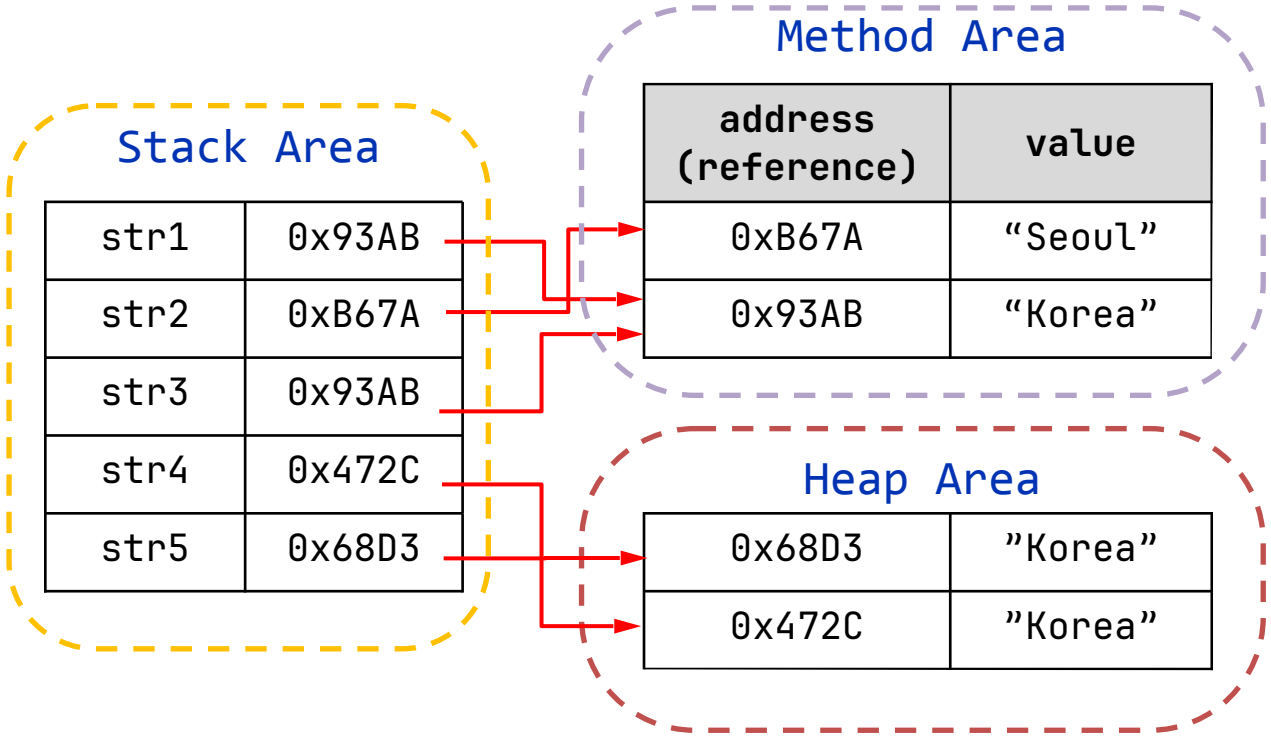


Memory Structure of JVM



Reference Type

- Reference Types
  - Having reference (address) of objects and arrays
- Ex)
  - String str1 = “Korea”;
  - String str2 = “Seoul”;
  - String str3 = “Korea”;
  - String str4 = new String(“Korea”);
  - String str5 = new String(“Korea”);



==, !=

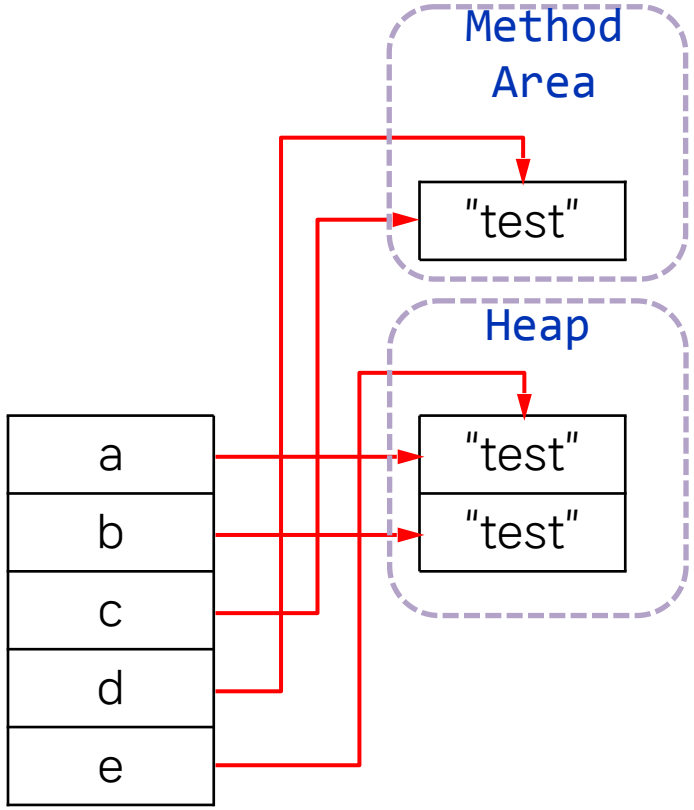
- For Primitive Type Variables
  - Test whether the values of two variables are the same or not
  - ex)

```
int x = 3;
int y = 2;
if (x == y || x != y + 2) { ... }
```

- For Reference Type Variables
  - Test whether the address (reference) of two variables are the same or not
  - That is, test whether the two variables are accessing the same object or not

==, !=

```java
public class TestTheSameReferences {
    public static void main(String[] args) {
        String a = new String("test");
        String b = new String("test");
        String c = "test";
        String d = "test";
        String e = a;
        System.out.println("a == b ? " + (a == b)); // false
        System.out.println("a == c ? " + (a == c)); // false
        System.out.println("a == d ? " + (a == d)); // false
        System.out.println("a == e ? " + (a == e)); // true
        System.out.println("b == c ? " + (b == c)); // false
        System.out.println("b == d ? " + (b == d)); // false
        System.out.println("b == e ? " + (b == e)); // false
        System.out.println("c == d ? " + (c == d)); // true
        System.out.println("c == e ? " + (c == e)); // false
    }
}
```



null

- Meaning: No object reference
  - ex)

```
String str = null;  // preferably initialized to null if there are no objects to reference
...
if (str == null)  { ... }  // if str doesn't reference any objects yet
else { ... }  // if str reference any valid object
```

- Dereferencing a ‘null’ reference causes a ‘NullPointException’
  - ex)

```
String s = null;  // no object yet
System.out.println(s.length()); // Throws NullPointerException
```

Example: Initialization as Null (1/2)

```
public class NullInitializationExample {
    public static void main(String[] args) {

        String str; // no initialization

        // COMPILE ERROR: "variable str might not have been initialized"

        if (str != null)
            System.out.println("Length of str: " + str.length());
        else
            System.out.println("str is null");

        str = null; // initialization

        if (str != null)
            System.out.println("Length of str: " + str.length());
        else
            System.out.println("str is null");
```

should be commented out

Example: Initialization as Null (2/2)

```
        str = "Hello"; // other case of initialization


        if (str ≠ null)
            System.out.println("Length of str: " + str.length());
        else
            System.out.println("This line will not be reached");
    }
}
```

```
str is null
Length of str: 5
```