Bonus

Referenstyper, objektstyper och stack och heap och GC

Referensvärde vs. objektvärde

Objektstyp

Referenstyp

```
List<String> list = new LinkedList<>();
LinkedList<String> linkedList = new LinkedList<>();

list.add(”world");
list.addFirst(”hello”); // Kompileringsfel!
linkedList.add("world");
linkedList.addFirst("hello");

System.out.println(list.getClass());
System.out.println(linkedList.getClass());
```

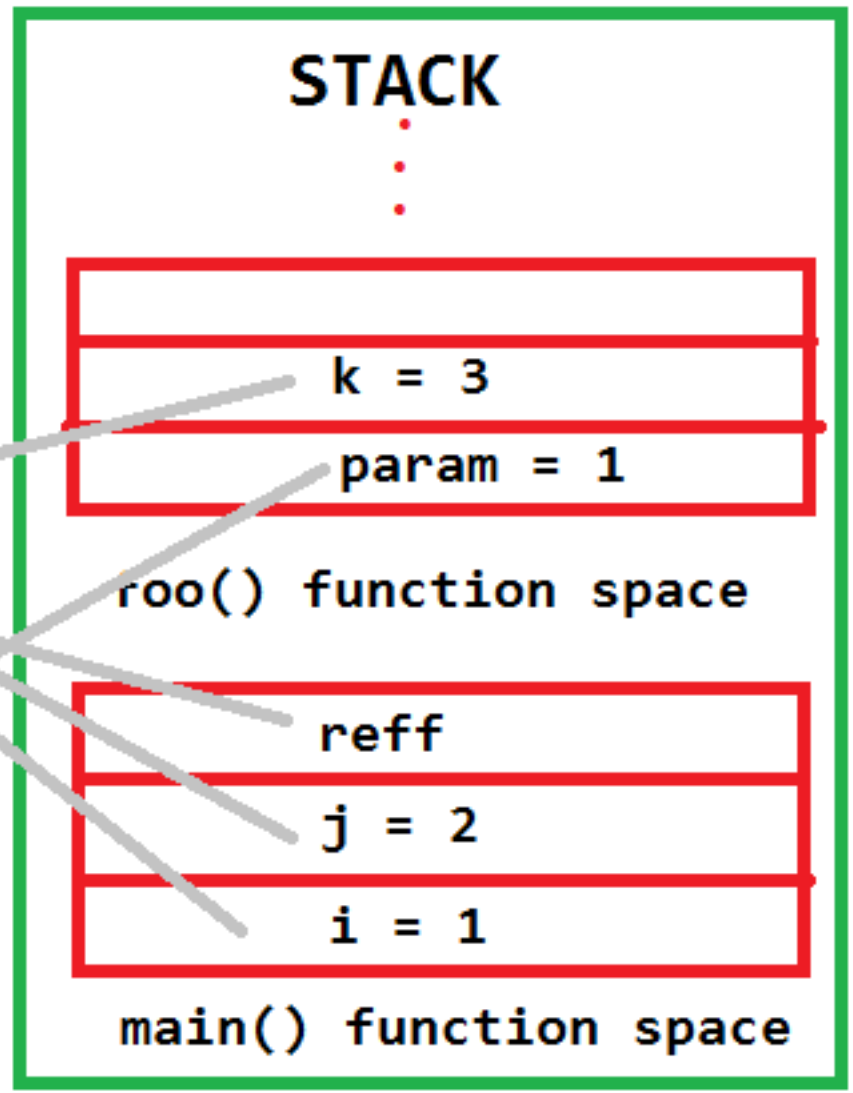
Oavsett referenstyp anropas objektstypens metod

Utskrift:
class java.util.LinkedList
class java.util.LinkedList

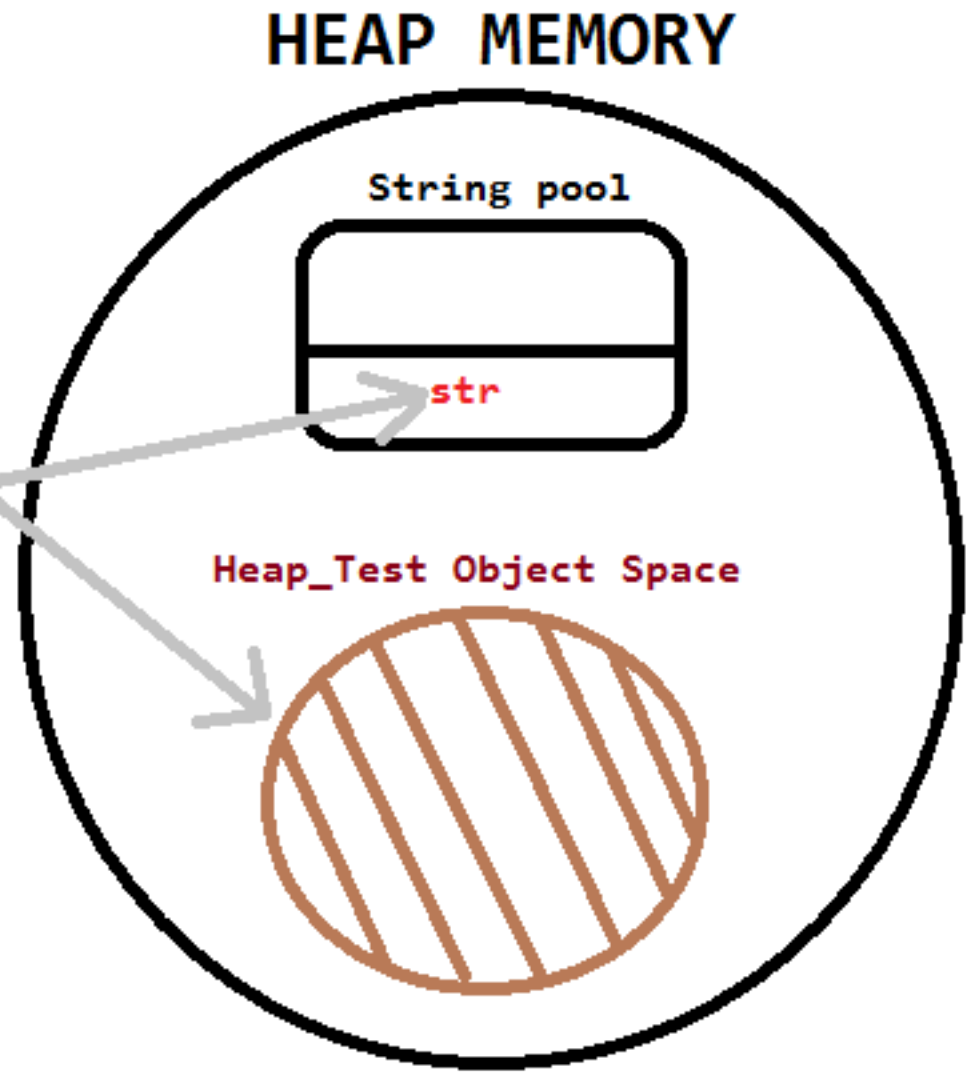
Primitivavärden vs objekt

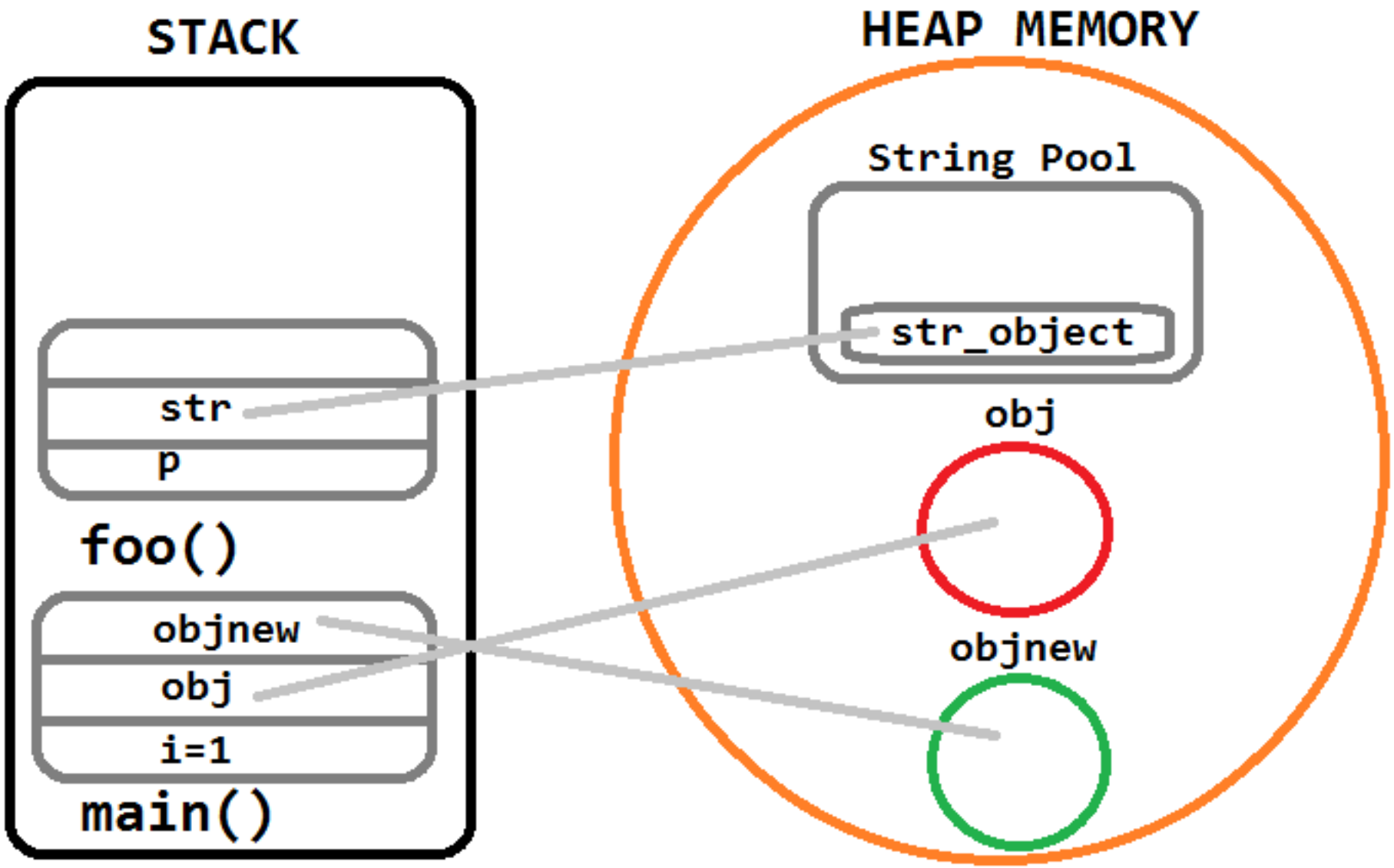
- Primitiva värden sparas på stacken och kan inte muteras (bara kopieras)
  - int a = 10; a.increaseValue(2) funkar ju inte!
  - a = a + 10; a skrivs över med värdet av a + 10
  - Är lokala i ett scope och vi kan inte skicka referenser till dem
- Objekt sparas på heapen och kan muteras
  - Häst h = new Häst();
  - h.setName(h); ändra hästens tillstånd
  - Objekten är inte lokala i sitt scope – dock kan deras referens vara det
  - När referensen inte ländre är nåbar raderas objektet ur heapen mha av sk. garbage collection

```java
public class Stack_Test {

 public static void main(String[] args) {

        int i=1;
        int j=2;

        Stack_Test reff = new Stack_Test();
        reff.foo(i);
    }

     void foo(int param) {
        int k = 3;
        System.out.println(param);
    }

}
```



```java
public class Heap_Test {

    public static void main(String[] args)
    {

        Heap_Test reff = new Heap_Test();

        reff.foo();
    }

     void foo() {

       String str = "Heap memory space";

        System.out.println(param);
    }

}
```




STACK
HEAP MEMORY
String Pool
str_object
str
p
foo()
obj
objnew
obj
i=1
main()
objnew

Garbage collection

- När objekt inte längre är nåbara – dvs det finns inte längre en referens i scope raderas det för att inte minnet ska ta slut

```
public static void main(String[] args) {
    List<Object> list = new LinkedList<>();
    f(list);
    System.out.println(list.get(0));
}

private static void f(List<Object> list) {
    String x = "hello world";
    List<String> l = new LinkedList<>();
    int i = 10;
    list.add(x);
}
```