References (Section 5.2)

Hsuan-Tien Lin

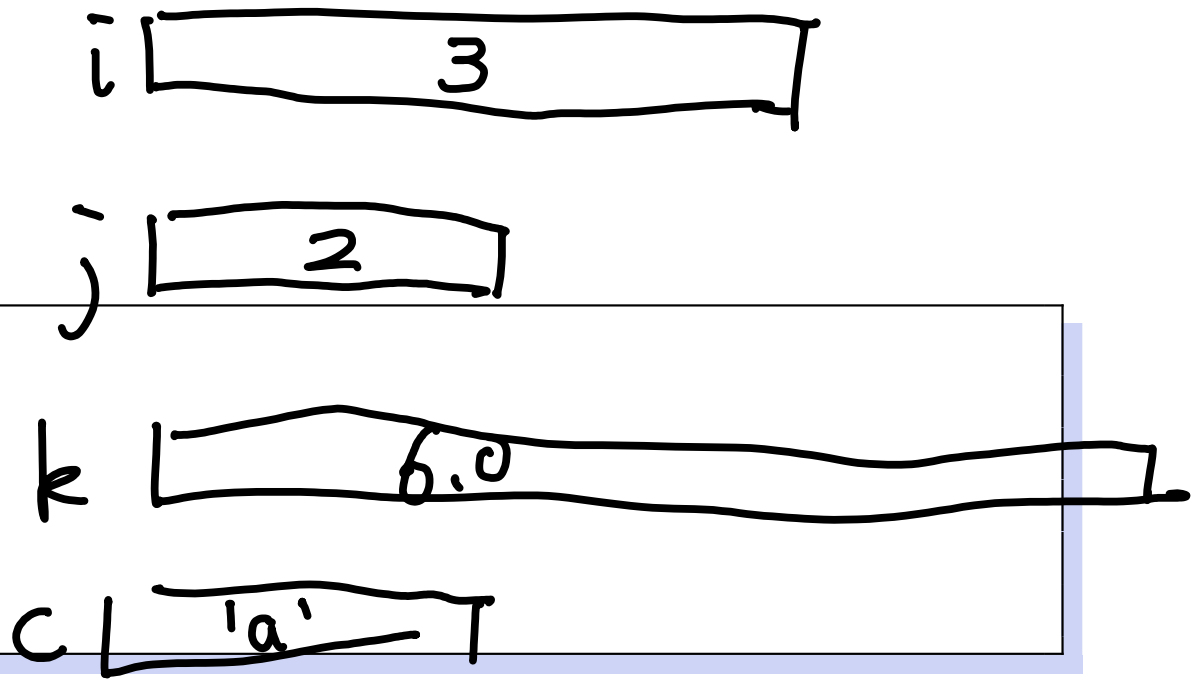
Deptartment of CSIE, NTU

OOP Class, March 15-16, 2010

Fun Time (1)

What happens in memory?

```
1  int i;
2  short j;
3  double k;
4  char c = 'a';
5  i = 3; j = 2;
6  k = i * j;
```



Life Cycle of a Primitive Variable (C/Java)

- declared and created

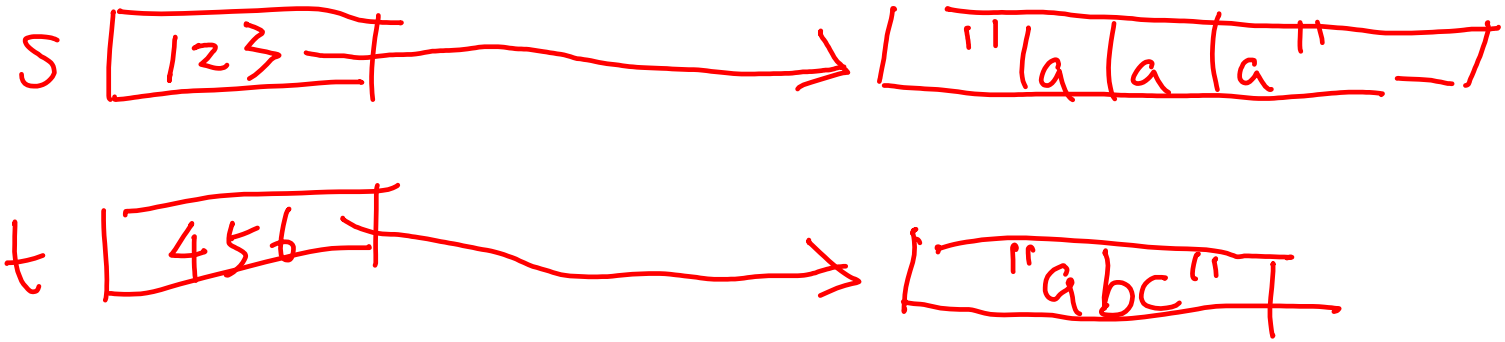
```
1  int count;
```

- used and modified

```
1  count += 1;
```

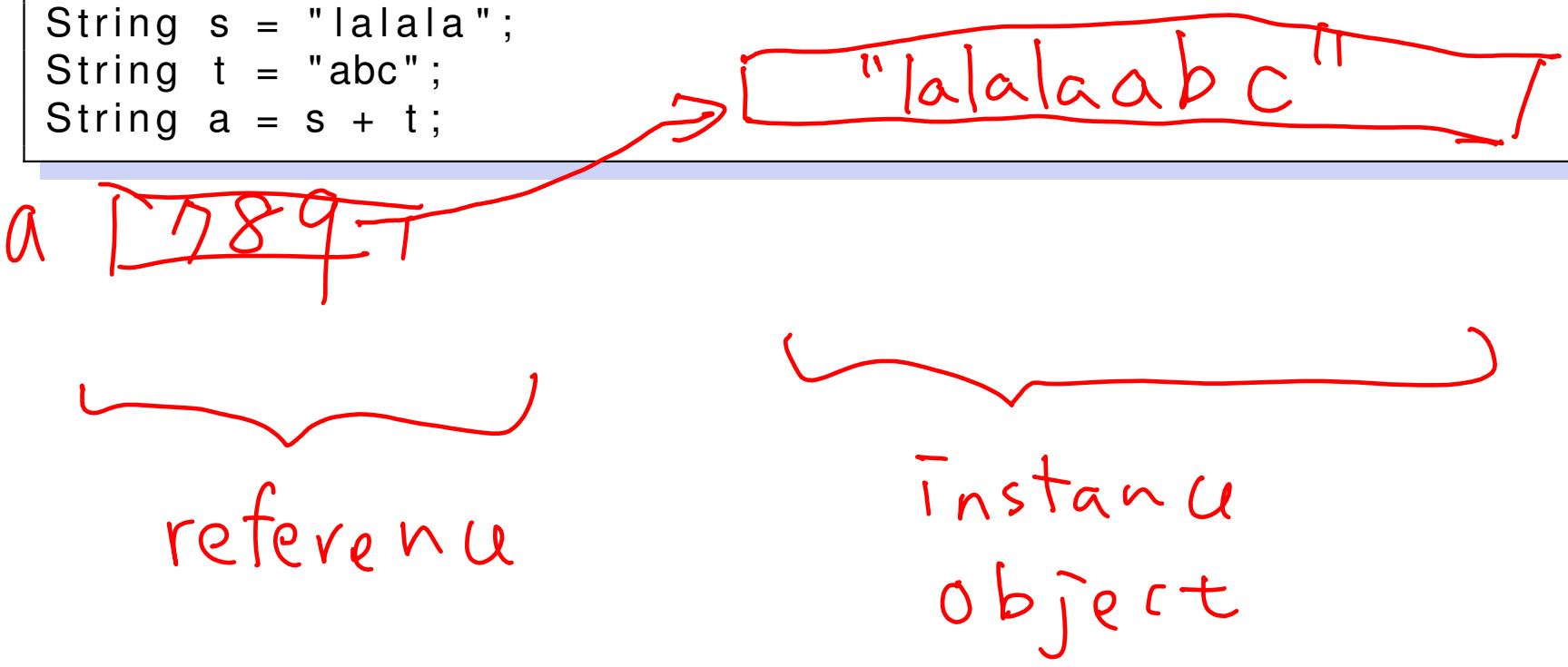
- destroyed
  –automatically (when out of scope)

Fun Time (2)

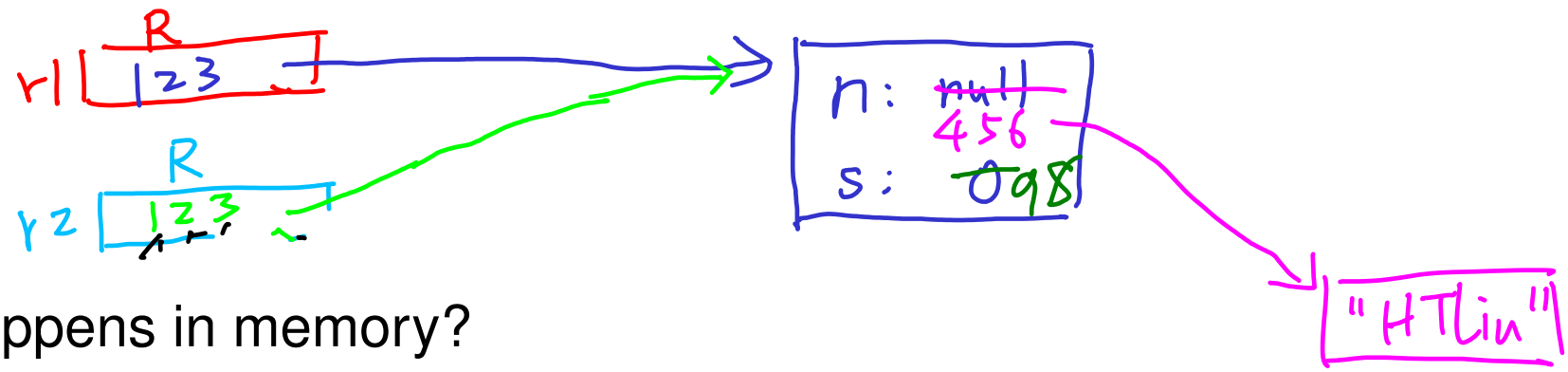


What happens in memory?

```
1 String s = "lalala";
2 String t = "abc";
3 String a = s + t;
```



Fun Time (3)

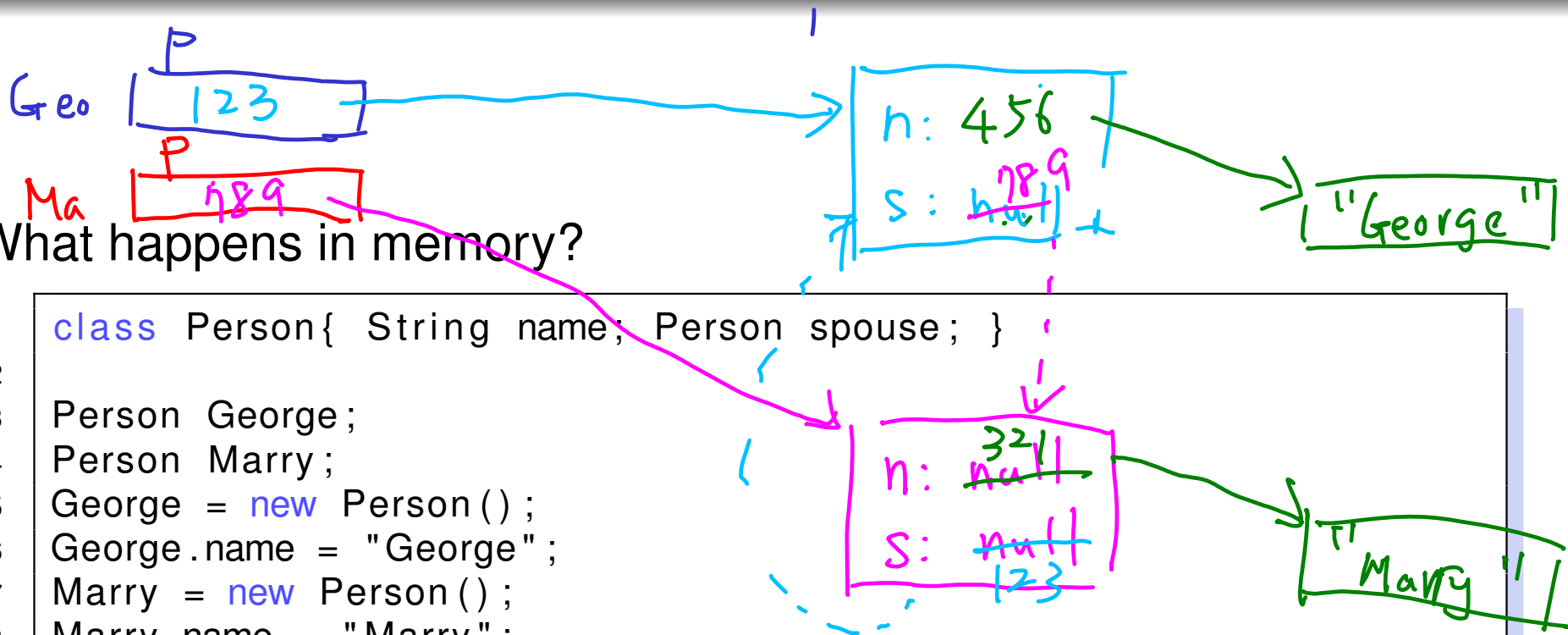


What happens in memory?

```
1 Record r1; //r1.name, r1.score
2 Record r2;
3 r1 = new Record();
4 r2 = r1;  //how many records are there?
5 r1.name = "HTLin";
6 r2.score = 98;
```

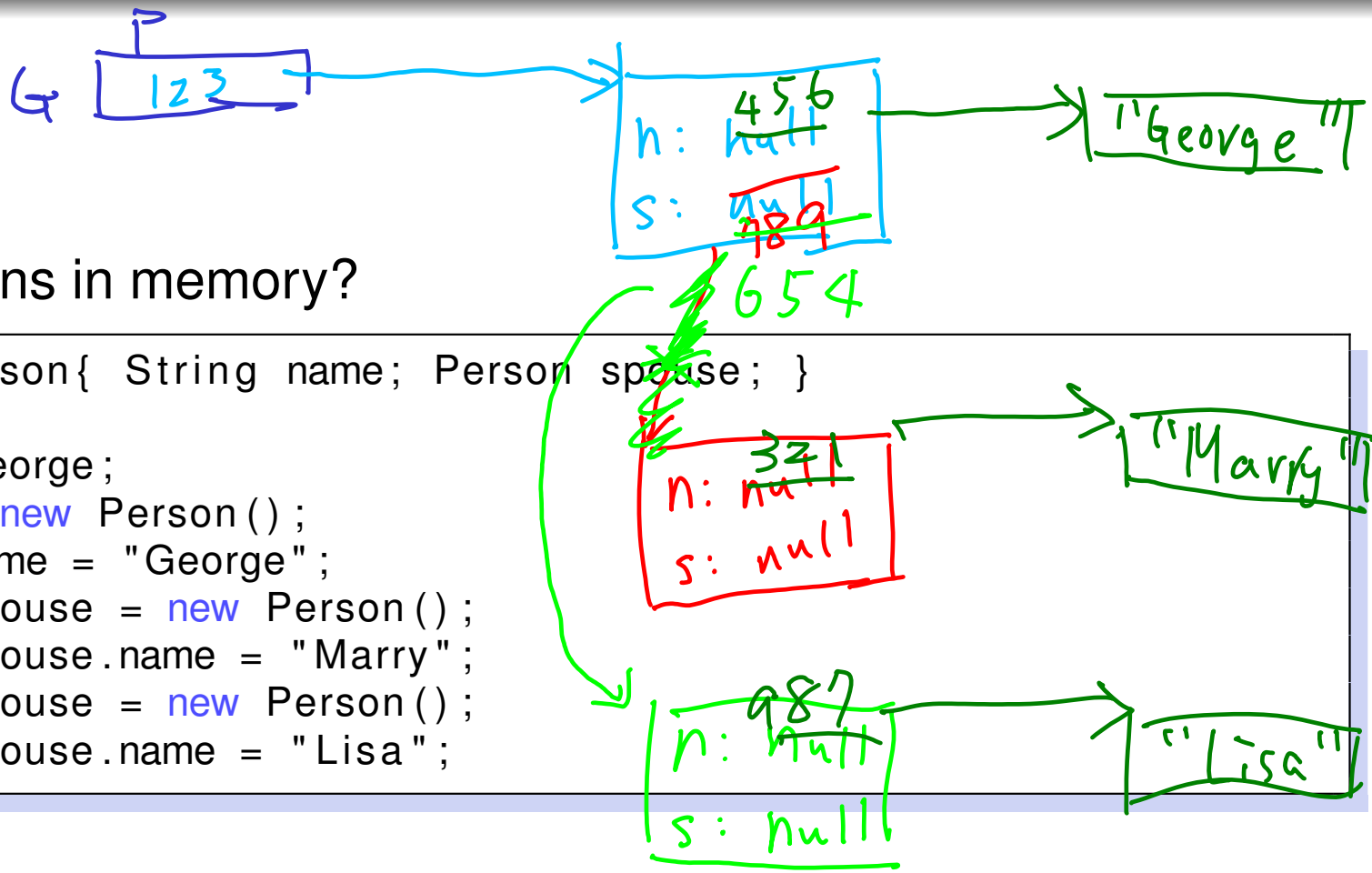
Fun Time (4)

What happens in memory?




```
1  class Person{ String name; Person spouse; }
2
3  Person George;
4  Person Marry;
5  George = new Person();
6  George.name = "George";
7  Marry = new Person();
8  Marry.name = "Marry";
9  Mary.spouse = George;
10 George.spouse = Marry;
```

Fun Time (5)



What happens in memory?

```
1  class Person{ String name; Person spouse; }
2
3  Person George;
4  George = new Person();
5  George.name = "George";
6  George.spouse = new Person();
7  George.spouse.name = "Marry";
8  George.spouse = new Person();
9  George.spouse.name = "Lisa";
```

Life Cycle of an Object Instance (Java)

- reference declared

```
1  Record r;
```

- instance created

```
1  r = new Record();
```

- used and modified

```
1  System.out.println(r.name);
```

- destroyed
  –automatically (when out of **use**)

Reference: Key Point

- a instance occupies a space in the memory;
  老太太住在屏東一個房子裡面
- reference (a.k.a. safe pointer): the “address” to the instance;
  用"海角七號"就可以找到老太太
- class-type variable: holds the reference;
  一個"信封"，上面寫著海角七號
- any operation on the instance goes thru the reference;
  要請老太太"回憶"時，拿個信封上寫"海角七號"，接著寫"回憶"，阿Ja就會使命必達了

```
老人  信封= new 老人(老太太的身家資料);
信封.回憶();
```

`null` Revisited (1/2)

```
1  class Record{
2    String name;
3    String ID;
4    int score;
5  }
6
7  public class RecordDemo{
8    public static void main(String[] arg){
9      Record r1 = new Record();
10     System.out.println(r1.score);
11     System.out.println(r1.name);
12   }
13 }
```

- `null`: Java's reserved word of saying "no reference"
- default initial value for extended types (if initialized automatically)
- `0`, `NULL`, anything equivalent to integer 0: C's way of saying "no reference"

`null` Revisited (2/2)

```
1  class Record{
2    String name;
3    String ID;
4    int score;
5  }
6
7  public class RecordDemo{
8    public static void main(String[] arg){
9      Record r1 = null;
10     System.out.println(r1.score);
11     System.out.println(r1.name);
12   }
13 }
```

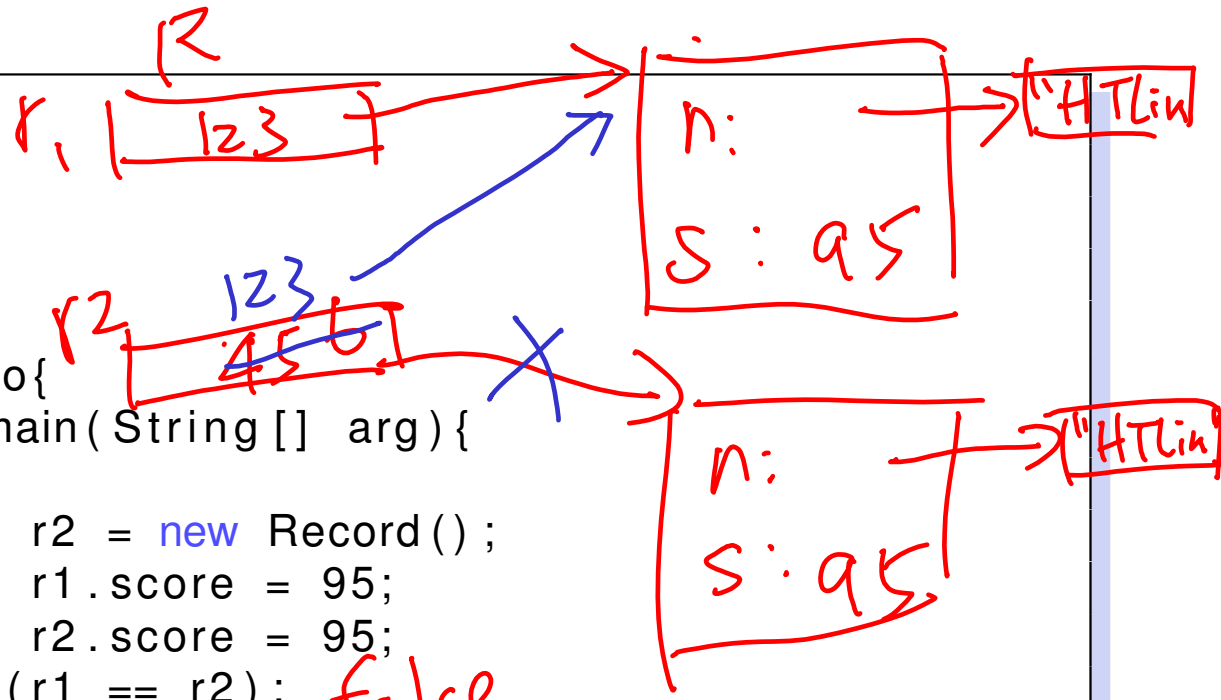
- null pointer exception (run time error): accessing the component of “no reference”

`null` Revisited: Key Point

`null`: Java's special way of saying "no reference"

Reference Equal (1/2)

```
1  class Record{
2    String name;
3    int score;
4  }
5
6  public class RecordDemo{
7    public static void main(String[] arg){
8      Record r1, r2;
9      r1 = new Record(); r2 = new Record();
10     r1.name = "HTLin"; r1.score = 95;
11     r2.name = "HTLin"; r2.score = 95;
12     System.out.println(r1 == r2);
13     r2 = r1;
14     System.out.println(r1 == r2);
15   }
16 }
```



- reference equal: comparison by “reference value”

Reference Equal (2/2)

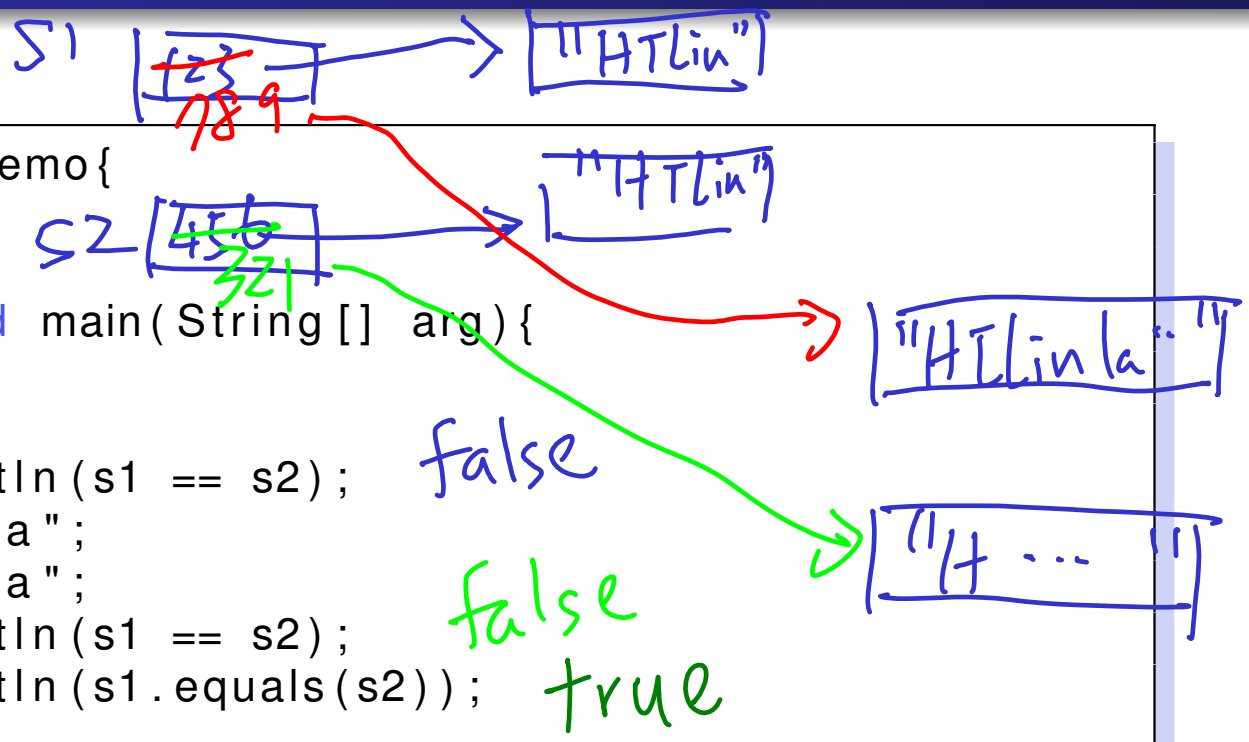
```
1  class Record{
2    String name;
3    int score;
4  }
5
6  public class RecordDemo{
7    public static void main(String[] arg){
8      Record r1, r2;
9      r1 = null; r2 = new Record();
10     System.out.println(r1 == r2);
11     r2 = r1;
12     System.out.println(r1 == r2);
13   }
14 }
```

- `null` does not equal non-null ..... o_O
- `null` equals `null` ..... O_o

Reference Equal: Key Point

==: reference equal rather than content equal for extended types

String Equal (1/1)




```
1  public class StringDemo{
2    static String s1;
3    static String s2;
4    public static void main(String[] arg){
5      s1 = "HTLin";
6      s2 = "HTLin";
7      System.out.println(s1 == s2);
8      s1 = s1 + "lalala";
9      s2 = s2 + "lalala";
10     System.out.println(s1 == s2);
11     System.out.println(s1.equals(s2));
12   }
13 }
```

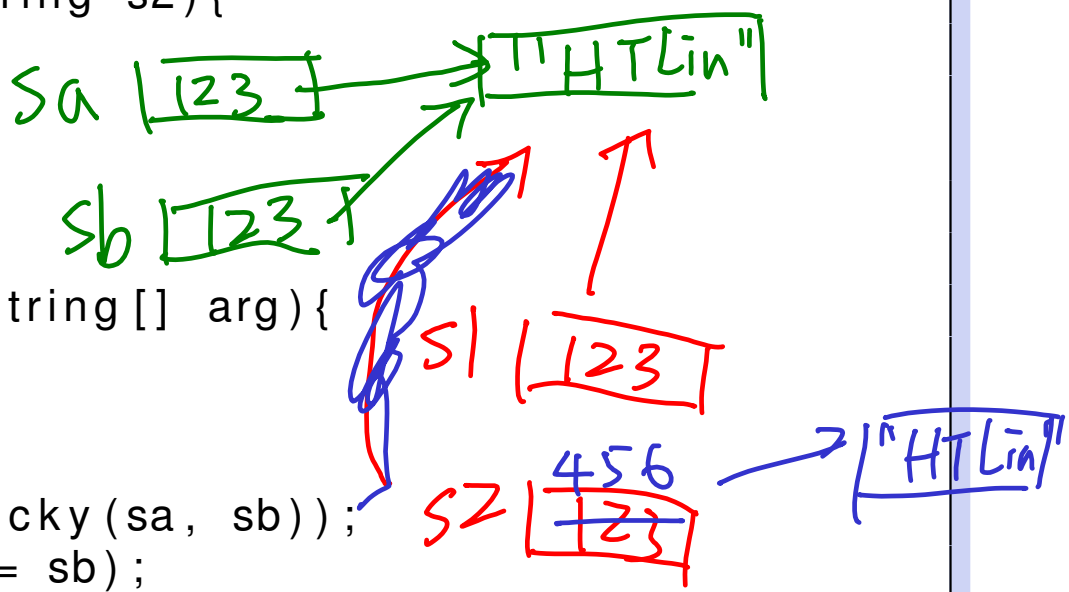
- first `true`: compiler allocates one constant string only
- second `false`: two different string references
- third `true`: an action (method) for content comparison

String Equal: Key Point

String `==`: still reference equal, use `.equals` if want content equal

Reference Argument/Parameter (1/3)

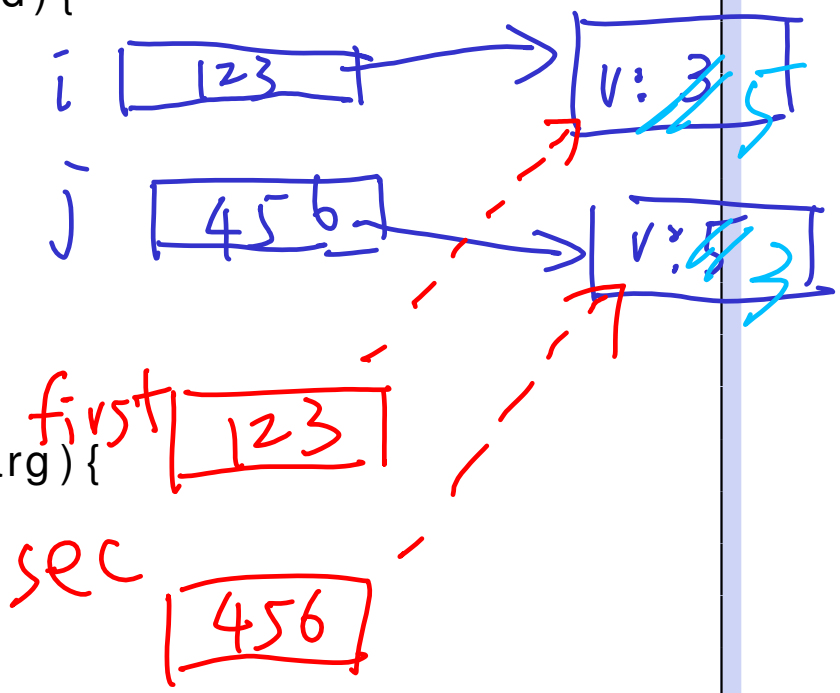
```
1  class Tool{
2    bool tricky(String s1, String s2){
3      s2 = s2 + "";
4      return (s1 == s2);
5    }
6  }
7  public class Demo{
8    public static void main(String[] arg){
9      Tool t = new Tool();
10     String sa = "HTLin";
11     String sb = sa;
12     System.out.println(t.tricky(sa, sb));
13     System.out.println(sa == sb);
14     System.out.println(t.tricky(sa + "", sb));
15   }
16 }
```



- reference parameter passing: again, value copying
- sa, sb copied to s1, s2
- s2 (reference) changed, sb didn't

Reference Argument/Parameter (2/3)

```
1  class myInt{int val; myInt(int v){val = v;}}
2  class Tool{
3    void swap(myInt first, myInt second){
4      int tmp = first.val;
5      first.val = second.val;
6      second.val = tmp;
7      System.out.println(first.val);
8      System.out.println(second.val);
9    }
10 }
11 public class Demo{
12   public static void main(String[] arg){
13     Tool t = new Tool();
14     myInt i = new myInt(3);
15     myInt j = new myInt(5);
16     t.swap(i, j);
17     System.out.println(i.val);
18     System.out.println(j.val);
19   }
20 }
```



- swapped as requested

Reference Argument/Parameter (3/3)

```
1  class myInt{int val; myInt(int v){val = v;}}
2  class Tool{
3    void swap(myInt first, myInt second){
4      myInt tmp = first;
5      first = second;
6      second = tmp;
7      System.out.println(first.val);
8      System.out.println(second.val);
9    }
10 }
11 public class Demo{
12   public static void main(String[] arg){
13     Tool t = new Tool();
14     myInt i = new myInt(3);
15     myInt j = new myInt(5);
16     t.swap(i, j);
17     System.out.println(i.val);
18     System.out.println(j.val);
19   }
20 }
```

自己畫畫看

- what happens?

Reference Argument/Parameter: Key Point

argument $\Rightarrow$ parameter: by reference copying
same for return value

this (1/1)

```
1  class Record{
2    int score;
3    void set_to(int score){ this.score = score; }
4    void adjust_score{ this.set_to(score+10); }
5  }
```

- which score? which set_to?
- `this`: my (the object's)

this: Key Point

`this`: the reference variable pointing to the object itself