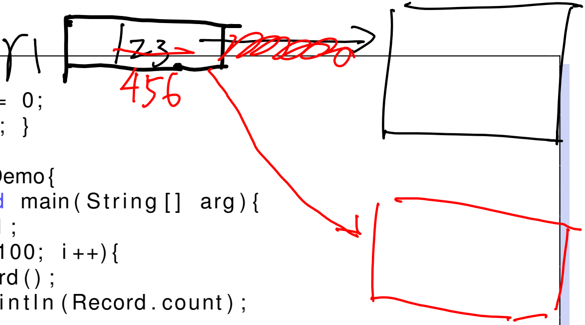
# More on Object Lifecycle

Hsuan-Tien Lin

Department of CSIE, NTU

OOP Class, March 22-23, 2010

# Garbage Collection (1/2)

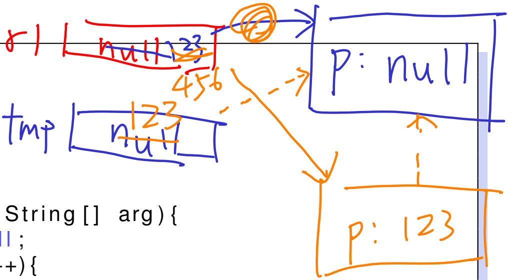


```
class Record{
  static int count = 0;
  Record(){ count++; }
}
public class RecordDemo{
  public static void main(String[] arg){
    int i; Record r1;
    for(i = 0; i < 100; i++){
      r1 = new Record();
      System.out.println(Record.count);
    }
  }
}
```

- 100 instances created, only 1 alive after the loop
- the other 99 memory slots: automatically recycled

# Garbage Collection (2/2)

```java
class Record{
  static int count = 0;
  Record prev;
  Record(){ count++; }
}
public class RecordDemo{
  public static void main(String[] arg){
    int i; Record r1 = null;
    for(i = 0; i < 100; i++){
      Record tmp = r1;
      r1 = new Record();
      r1.prev = tmp;
      System.out.println(Record.count);
    }
  }
}
```



- 100 instances created, all of them alive

# Garbage Collection: Key Point

Garbage Collection: when a memory slot becomes an orphan (and) system in need of memory

# Finalizer (1/2)

```java
class Record{
  static int mem = 0;
  Record(){ mem += 10; }
  void when_truck_comes(){ mem -= 10; }
}
public class RecordDemo{
  public static void main(String[] arg){
    int i; Record r1;
    for(i = 0; i < 100; i++){
      r1 = new Record();
      System.out.println(Record.mem);
    }
  }
}
```

- finalizer: something you want to do when truck comes
- calculate memory usage, write something back (say, on BBS), ...

# Finalizer (2/2)

```java
class Record{
  static int mem = 0; static int count = 0;
  int id;
  Record(){ mem += 10; count++; id = count; }
  protected void finalize() throws Throwable{
    System.out.print(id);
    System.out.println(",␣Good␣Bye!");
    mem -= 10;
  }
}
public class RecordDemo{
  public static void main(String[] arg){
    int i; Record r1 = null;
    for(i = 0; i < 100; i++){
      Record tmp = r1; r1 = new Record();
      System.out.println(Record.mem);
    }
  }
}
```

- GC: no guarantee on when the truck comes
- if JVM halts before truck comes, even no finalizer calls

# Finalizer: Key Point

finalizer:

a mechanism to let the instance say goodbye

# Object Lifecycle (1/1)

```
class Record{
  int score;
  Record(int init_score){ score = init_score; }
  protected void finalize() throws Throwable{ }
}
public class RecordDemo{
  public static void main(String[] arg){
    Record r; //reference declared
    Record r2;//reference declared
    r = new Record(60); //memory allocated (RHS)
                        //and constructor called
                        //reference assigned (LHS)
    r2 = r;             //reference copied
    r.score = 3;        //instance content accessed
    r.show_score();     //instance action performed
    r2 = null; r = null;//memory slot orphaned
    //....
                        //finalizer called
                        //or JVM terminated
  }
}
```

# Object Lifecycle: Key Point

we control birth, life, death, funeral design, but not the exact funeral time