Class versus Instance (Section 5.1)

Hsuan-Tien Lin

Department of CSIE, NTU

OOP Class, March 22-23, 2010

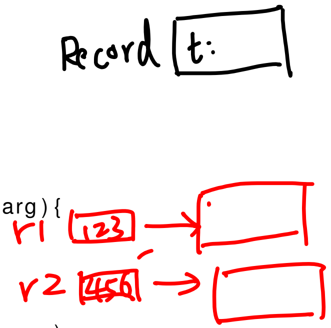
Static Variables (1/2)

```
class Record{
  int total_rec;
  public Record(){
    total_rec += 1;
  }
  public void show_total_rec(){
    System.out.println(total_rec);
  }
}
public class RecordDemo{
  public static void main(String[] arg){
    Record r1 = new Record();
    r1.show_total_rec();
    Record r2 = new Record();
    r2.show_total_rec();
  }
}
```

- no shared space to store the total records

Static Variables (2/2)

```java
class Record{
  static int total_rec = 0;
  public Record(){ total_rec++; }
  public void show_total_rec(){
    System.out.println(total_rec);
  }
}
public class RecordDemo{
  public static void main(String[] arg){
    Record r1 = new Record();
    r1.show_total_rec();
    Record r2 = new Record();
    r2.show_total_rec();
    System.out.println(Record.total_rec);
  }
}
```



r1 . total_rec

- static: shared between all X-type instances
- like a global variable within the scope of the class
- use scarcely

Static Variables: Key Point

`static` variable:
of the **class** (shared), not of an instance

Static Final Variables (1/3)

```java
class Circle{
  double r;
  public Circle(double radius){
    r = radius;
  }
  public void show_area(){
    System.out.println(3.141592654 * r * r);
  }
  public void show_cir(){
    System.out.println(2.0 * 3.141592654 * r);
  }
}
public class CircleDemo{
  public static void main(String[] arg){
    Circle c = new Circle(3);
    c.show_area();
  }
}
```

- typing many 3.141592654 looks silly
- 3.141592654 does not need to be per-instance

Static Final Variables (2/3)

```java
class Circle{
  static double p = 3.141592654;
  double r;
  public Circle(double radius){ r = radius; }
  public void show_area(){
    System.out.println(p * r * r);
  }
  public void show_cir(){
    System.out.println(2.0 * p * r);
  }
}
public class CircleDemo{
  public static void main(String[] arg){
    Circle c = new Circle(3); c.show_area();
    c.p = 10; c.show_area();
  }
}
```

- prevention: don't use names `r`, `p`
- prevention: don't allow modify `p`

Static Final Variables (3/3)

```
class Circle{
  static final double p = 3.141592654;
  double r;
  public Circle(double radius){ r = radius; }
  public void show_area(){
    System.out.println(p * r * r);
  }
  public void show_cir(){
    System.out.println(2.0 * p * r);
  }
}
public class CircleDemo{
  public static void main(String[] arg){
    Circle c = new Circle(3);
    c.show_area();
    c.p = 10; //a typo here
  }
}
```

- static final: Java's way of saying constant (over the class)

Static Final Variables: Key Point

`static final` variable: **constant**

Static Variables Revisited (1/1)

```
class Record{
  static int total_rec = 0;
  int id;
  public Record(){ id = total_rec++;}
}
public class RecordDemo{
  public static void main(String[] arg){
    Record r1 = new Record();
    Record r2 = null;
    Record r3 = new Record();
    System.out.println(r1.total_rec);
    System.out.println(r2.total_rec);
    System.out.println(Record.total_rec);
    System.out.println(r1.id);
    System.out.println(r2.id);
    System.out.println(Record.id);
  }
}
```

1 $\boxed{2}$

$\boxed{2}$ null haha ohoh

$\boxed{2}$

$\boxed{0}$ 1

1 null ohohoh

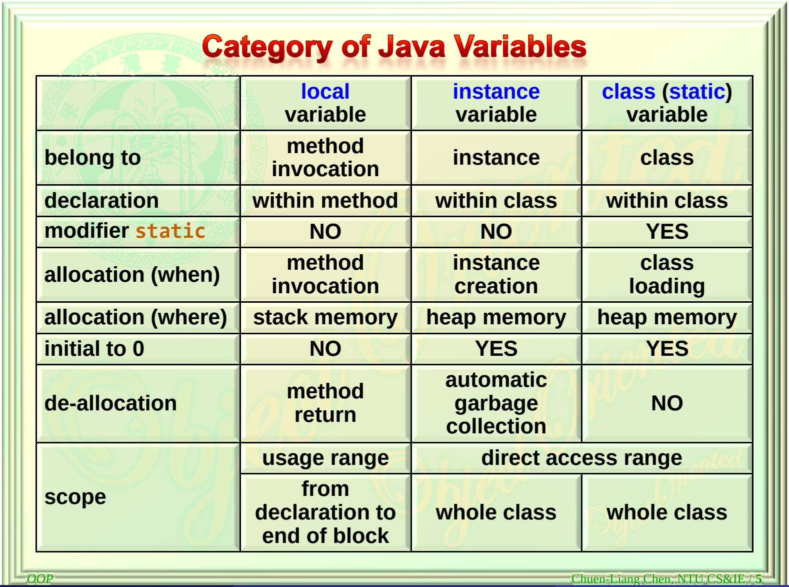
$\boxed{\text{haha}}$

- `r2.total_rec` ⇒ `Record.total_rec` in **compile time**

Static Variables Revisited: Key Point

`static` variable:
of the **class** (shared), not of an instance

Java Variables Revisited (courtesy of Prof. Chuen-Liang Chen)



Category of Java Variables

	local variable	instance variable	class (static) variable
belong to	method invocation	instance	class
declaration	within method	within class	within class
modifier `static`	NO	NO	YES
allocation (when)	method invocation	instance creation	class loading
allocation (where)	stack memory	heap memory	heap memory
initial to 0	NO	YES	YES
de-allocation	method return	automatic garbage collection	NO
scope	usage range	direct access range	
	from declaration to end of block	whole class	whole class

OOP Chuen-Liang Chen, NTU CS&IE / 5

Static Methods (1/2)

```
class myMath{
  double mean(double a, double b){
    return (a + b) * 0.5;
  }
}
public class MathDemo{
  public static void main(String[] arg){
    double i = 3.5;
    double j = 2.4;
    myMath m = new MyMath();
    System.out.println(m.mean(i, j));
  }
}
```

- new a `myMath` instance just for computing `mean`
  –lazy people don't want to do so

Static Methods (2/2)

```
class myMath{
  static double mean(double a, double b){
    return (a + b) * 0.5;
  }
}
public class MathDemo{
  public static void main(String[] arg){
    double i = 3.5;
    double j = 2.4;
    System.out.println(myMath.mean(i, j));
    System.out.println((new myMath() ).mean(i, j));
  }
}
```

myMath tmp = new myMath();
System....(tmp.mean(i, j));

- make the method a `static` (class) one
  –no need to new an instance
- similar to static variable usage   System....(myMath.mean(i, j));

Static Methods: Key Point

`static` method:
associated with the **class**,
no need to create an instance

Use of Static Methods (1/2)

```java
public class UtilDemo{
  public static void main(String[] arg){
    System.out.println(Math.PI);
    System.out.println(Math.sqrt(2.0));
    System.out.println(Math.max(3.0, 5.0));
    System.out.println(Integer.toBinaryString(15));
  }
}
```

- commonly used as utility functions (so don’t need to create instance)
- main is static (called by classname during ’`java className`’)
- tools for other static methods

Use of Static Methods (2/2)

```
class Record{
  static int total_rec = 0;
  Record(){ total_rec++; }
  static void show_total_rec(){
    System.out.println(total_rec);
  }
}
public class RecordDemo{
  public static void main(String[] arg){
    Record r1 = new Record();
    Record.show_total_rec();
  }
}
```

- class related actions rather than instance related actions

Use of Static Methods: Key Point

`static` method:

- compile time determined
- per class
- sometimes useful

```
class Record{
  String name; int score;
  public static void main(String[] arg){
    Record r = new Record();
    r.name = "lalala";
    r.score = 60;
  }
}
```