

# Constructor (Section 4.4)

Hsuan-Tien Lin

Department of CSIE, NTU

OOP Class, March 15-16, 2010

# Constructor (1/3)

```
1  class Record{
2      String name;
3      int score;
4  }
5  // ...
6  r = new Record();
```

- the `new` operator allocates memory for the instance
- often you will do this:

```
1  r = new Record();
2  r.name = "HTLin";
3  r.score = 99;
```

94

- out of laziness, you want to do this:

```
1  r = new Record("HTLin", 90);
```

## How?

## Constructor (2/3)

```
1  class Record{
2      String name;
3      int score;
4      public Record(String init_name,
5                      int init_score){
6          name = init_name;
7          score = init_score;
8      }
9  }
10 // ...
11 r = new Record("HTLin", 90);
```

## Constructor (3/3)

- constructor: called by `new` to **initialize**
- name: same as class name
- the `public` constructor
- default constructor (if you didn't write any code): same as

```
1 public Record() {  
2 }
```

- constructor without argument (“replace” the default one):

```
1 public Record() {  
2     score = 60;  
3 }
```

# Constructor: Key Point

a special method to initialize the instance during `new`

## More on Constructors (1/3)

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

hahaha

- default value (when `new`): 0/false/null
- default constructor when no self-defined constructor

## More on Constructors (2/3)

```
1  class Record{
2      String name; int score;
3      Record(int init_score){score = init_score;}
4  }
5  public class RecordDemo{
6      public static void main(String[] arg){
7          Record r1 = new Record(60);
8          Record r2 = new Record();
9          System.out.println(r1.score);
10         System.out.println(r2.score);
11     }
12 }
```

- if there is self-defined constructor, no default one
- self-defined constructor: same-name, no return value (but not void)

## More on Constructors (3/3)

```
1  class Record{
2      String name; int score;
3      Record(int init_score){score = init_score;}
4      Record(){ Record(40);}
5  }
6  public class RecordDemo{
7      public static void main(String[] arg){
8          Record r1 = new Record(60);
9          Record r2 = new Record();
10         System.out.println(r1.score);
11         System.out.println(r2.score);
12     }
13 }
```

- can **overload**: same name, different parameters
- can call other constructors to help initialize



## More on Constructors: Key Point

often better to use self-defined and overloaded constructors to help initialize