Reference Assignment and Inheritance (1/4)

```
1  class Student{ int ID; String name; }
2  class GodStudent extends Student{ Award[] president_awards; }
3
4  public class StudentDemo{
5    public static show_student_id(Student s){
6      System.out.println(s.ID);
7    }
8    public static void main(String[] argv){
9      GodStudent htlin = new GodStudent();
10     show_student_id(htlin);
11     Student CharlieL = new Student();
12     show_student_id(CharlieL);
13   }
14 }
```

- htlin refers to an instance of GodStudent (for sure!)
- htlin refers to an instance of Student as well
- one instance, many coherent types (in argument passing, return value, etc.)

Reference Assignment and Inheritance (2/4)

```
1  class Student{ int ID; String name; }
2  class GodStudent extends Student{ Award[] president_awards; }
3
4  public class StudentDemo{
5    public static void main(String[] argv){
6      GodStudent htlin = new GodStudent();
7      //I want to be a usual student
8      Student usualstudent = htlin;
9      Student anotherusualstudent = new Student();
10   }
11 }
```

- if “copying assignment”, a copy of htlin can be a usual student
- but “reference assignment” in Java, how can htlin be usual?
- mechanism?

Reference Assignment and Inheritance (3/4)

```
1  class Student{ int ID; String name; }
2  //takes 4 + (8) bytes
3  class GodStudent extends Student{ Award[] president_awards; }
4  //takes 4 + (8) + (8) bytes
5
6  /** excluding some other information, much like
7  class GodStudent{
8    //first 4 + (8) bytes (for Student)
9    int ID;
10   String name;
11   //last (8) bytes (for GodStudent)
12   Award[] president_awards;
13 }//takes 4 + (8) + (8) bytes
14 **/
```

- one possible mechanism for single inheritance (Java!!): shared prefix (virtually)

Reference Assignment and Inheritance (4/4)

```
1  class Student{ int ID; String name; }
2  //class Student{ int 000; reference 004; }
3  class GodStudent extends Student{ Award[] president_awards; }
4  //class GodStudent{ int 000; reference 004; reference 012; }
```

- one possible mechanism: variable name $\Rightarrow$ a single number when class **loads**
- (after instance type check) objects can just safely access memory contents by numbers

Reference Assignment and Inheritance: Key Point

a simple run-time mechanism (shared prefix):
ancestor first, descendant last