CISC 3115 TY3

C12a: The Object Superclass and Selected Methods

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Outline

- The Object class and its methods
  - toString()
  - equals and its contract
- The protected visibility modifier
- The final modifier
- ArrayList, arrays, and Collections

The Object Class

- At the top of the Java class hierarchy tree is the java.lang.Object class
- Every class in Java is descended from the java.lang.Object class.
- Even if no inheritance is specified when a class is defined, the superclass of the class is actually Object.

```
public class Circle {
  ...
}
```

Equivalent

```
public class Circle extends Object {
  ...
}
```

The Superclass: The Object Class

- The Object class is a superclass of all Java classes
  - Every class you use or write inherits the instance methods of the Object class
  - You may override the methods with an implementation that is specific to your class.

The toString() Method in Object

- The toString() method returns a string representation of the object.
  - The default implementation

```
public String toString() {
    return getClass().getName() + "@" + Integer.toHexString(hashCode());
}
```

  - returns a string consisting of
    - a class name of which the object is an instance,
    - the at sign (@), and
    - a number representing this object.

Example: The toString() Method

- Example: try these statements

```
Loan loan = new Loan();
System.out.println(loan.toString());
```

Overriding the toString() Method

- The toString() method is often overridden.

Questions?

- The Object class and its toString() method

Comparing Objects

- Given two reference variables v1 and v2, you may do comparison as follows,
  - v1 == v2
  - v1.equals(v2)
- where the equals method is defined in the Object class

v1 == v2

- compares the references held in v1 and v2 and determine whether they are identical.

Remark: The == Operator

- It is used for comparing
  - two primitive data type values
  - or for determining whether two objects have the same references.

v1.equals(v2)

- It depends on the implementation of the equals method
- The equals method is defined in the Object class with the following implementation

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

Example: Comparing Students

- Consider a Student class

```
public class Student {
    private int studentId;
    private String name;
    public Student(int sid, String name) { …}
    …
}
```

- What do think you <u>should</u> get?

```
Student s1 = new Student(100, "John Doe");
Student s2 = new Student(100, "John Doe");
System.out.println(s1.equals(s2));
```

Overriding the Equals Method

- We override the equals method in the Student class

```
public boolean equals(Object theOther) {
   if (theOther instanceof Student) {
     return id == ((Student)theOther).id &&
name.equals(((Student)theOther).name);
   } else {
     return false;
   }
 }
```

Am I Overriding it?

- How about this?

```
public boolean equals(Student theOther) {
   if (student != null) {
     return id == theOther.id && 
name.equals(theOther.name);
   } else {
     return false;
   }
 }
```

No. You Aren't

- These are two different methods

```
boolean equals(Student theOther) {…}
boolean equals(Object theOther) {…}
```

Remark: The equals Method

- It is intended to test whether two objects have the same contents, provided that the method is overridden in a class, a subclass of Object.
- The == operator is stronger than the equals method, in that the == operator checks whether the two reference variables refer to exactly the same object in the memory (the heap).

Enforcing the Contract

- The API documentation states,

  "Note that it is generally necessary to override the hashCode method whenever this method is overridden, so as to maintain the general contract for the hashCode method, which states that equal objects must have equal hash codes."

Questions?

- Every class in Java is a descendent of the Object class
- To compare two objects, we generally need to override the equals method
- What is the intended difference between the == operator and the equals method?
- How do we properly override the equals method?