Advanced Java Programming

Part 4: The `static` and `final` keywords, useful methods to define for your classes: `toString()`, `equals()`

James Tam

We Create Several Sheep

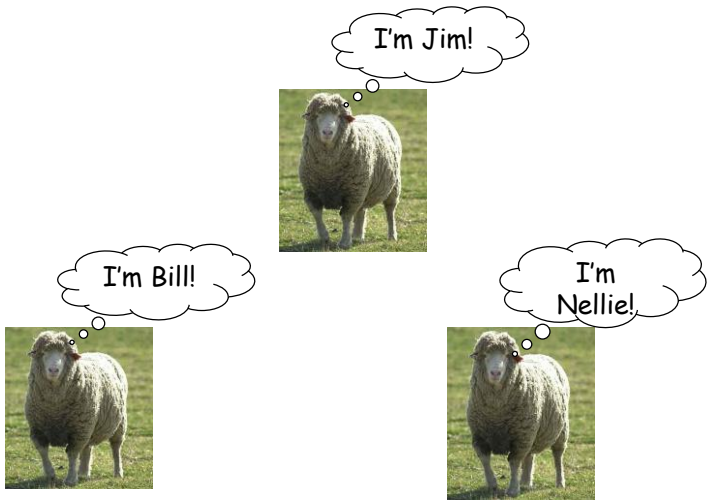


Image copyright unknown

James Tam

Question: Who Tracks The Size Of The Flock?

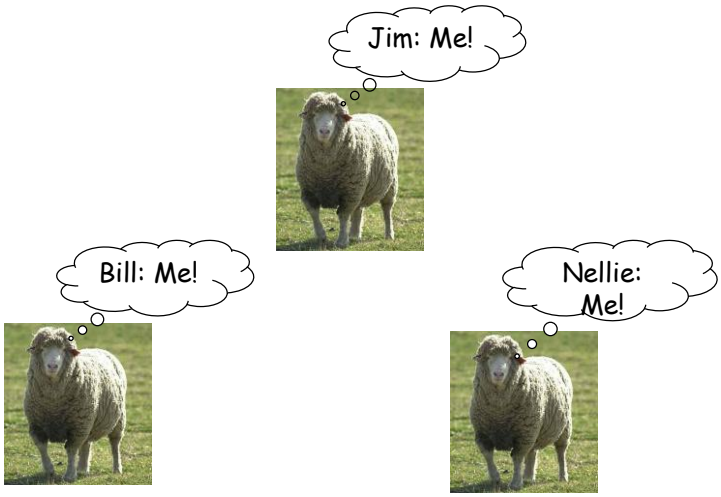


Image copyright unknown

James Tam

Answer: None Of The Above!

- Information about all instances of a class should not be tracked by an individual object.
- So far we have used instance fields.
- Each *instance* of an object contains *it's own set of instance fields* which can contain information unique to the instance.

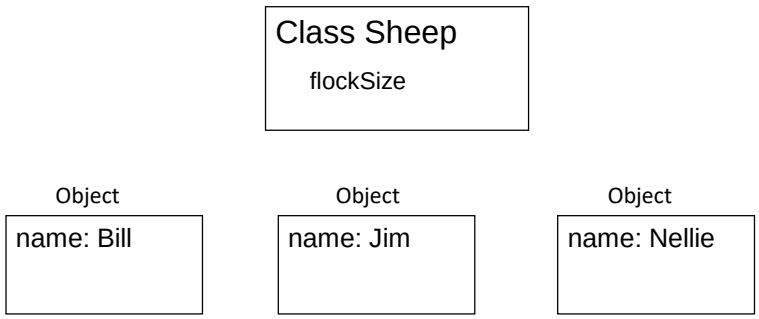
```
public class Sheep
{
     private String name;
       ...
}
```

Object	Object	Object
name: Bill	name: Jim	name: Nellie

James Tam

The Need For Static (Class Attributes)

- Static fields: One instance of the attribute exists *for the class* (not one attribute for each instance of the class)
- JT's note: in Java `static` DOES NOT specify unchanging (constant)
  - Reminder: the keyword '`final`' signifies constant (unchanging)



James Tam

`Static` (Class) Methods

- **New term, static methods**: Are associated with the class as a whole and not individual instances of the class.
  - Can be called without having an instances (because it's called through the class name not a reference/instance name).
  - Instance method:

```
Scanner in = new Scanner(System.in);
in.nextInt();   // referenceName.method()
```

  - Class Method:

```
double squareRoot = Math.sqrt(9);  // ClassName.method()
```

- Typically implemented for classes that are never instantiated e.g., class `Math`.

James Tam

Defining Static Methods/Attributes

Format:

```
<Access permission> static <attribute or method name>
```

Example:

```
class Sheep
{
    private static int flockSize = 0;

}
```

James Tam

Accessing Static Methods/Attributes

- Inside the class definition just specify the name of the attribute or method.

Example:

```
class Sheep
{
    private static int flockSize = 0;

    public Sheep()
    {
        flockSize++;
    }
}
```

James Tam

Accessing Static Methods/Attributes (2)

- Outside the class definition preface the attribute or method with the name of the class.

Format:

```
<Class name>.<attribute or method name>
```

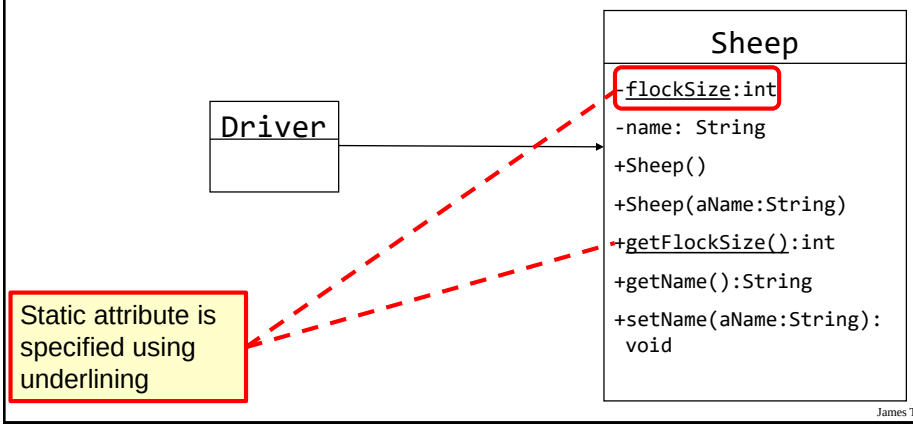
Example:

```
Sheep.getFlockSize();
```

James Tam

Static Data And Methods: UML Diagram

- **Name of the folder containing the complete example** : `10classAttributes`



James Tam

Static Data And Methods: The Driver Class

```
public class Driver
{
    public static void main(String [] args) {
        System.out.println();
        System.out.println("You start out with " +
                           Sheep.getFlockSize() +
                          " sheep");
        System.out.println("Creating flock...");
        Sheep nellie = new Sheep("Nellie");
        Sheep bill = new Sheep("Bill");
        Sheep jim = new Sheep();
        System.out.println("Current count " +
                           Sheep.getFlockSize());
    }
}
```

James Tam

Static Data And Methods: The Sheep Class

```
public class Sheep
{
    private static int flockSize = 0;
    private String name;

    public Sheep() {
        flockSize++;
        name = "No name";
    }
    public Sheep(String aName) {
        flockSize++;
        setName(aName);
    }

    public static int getFlockSize() { return flockSize; }
    public String getName() { return name;}
    public void setName(String newName) { name = newName; }
}
```

James Tam

Rules Of Thumb: Instance Vs. Class Attribute

- Reminder:
  - Instance attribute:
    - Static keyword is not used
    - There is one instance for each object created
    - E.g., `class Person { private int age; }`
- **New term**:
  - Class attribute:
    - Requires the static keyword
    - There is one instance for the entire class
    - E.g., `class Person { private static int numberPeople; }`
- Rules of thumb:
  - Make it an instance field if the data can vary between instances e.g., age, height, weight
  - Make it a class field if the data relates to all instances e.g., number of objects created.
    - Possibly it may apply if no instances will be created e.g., a debug flag to specify the mode that the program is operating under

James Tam

Rule Of Thumb: Instance Vs. Class Methods

- Reminder:
  - Instance method e.g.,

```
class Person { private int age = 0;
    public void haveBirthDay() { age++; }
}
```

  - Class method e.g.,

```
class Math {
    public static double square(double num) {return(num*num);
} }
```

James Tam

Rule Of Thumb: Instance Vs. Class Methods (2)

- Rule of thumb
  - Static methods
    - If a method can be invoked regardless of the number of instances that exist (e.g.., the method can be run when there are no instances) then it probably should be a static method.
    - If it never makes sense to instantiate an instance of a class then the method should probably be a static method.
      E.g., the class doesn't have any variable attributes only static constants such as class `Math` no objects are instantiated (more coverage later)
  - Non static methods
    - If the above rules don't apply then the method should likely be an instance method e.g., the method operates on an instance field.

James Tam

Universally Accessible Constants

- What you currently know
  - How to declare constants that are local to a method

```
class Driver {
    main() {
        final int A_CONST = 10;
    }
}
```

- If you need constants that are accessible throughout your program then declare them as **class constants**.

James Tam

Declaring Class Constants

- **Format:**

```
public class <class name>
{
    public final static <type> <NAME> = <value>;
}
```

- **Example:**

```
public class Person
{
    public final static int MAX_AGE = 144;
}
```

- Notes:
  - The keyword "final" signifies something that cannot change (a constant)
  - Because MAX_AGE is a constant the access level can be public.

James Tam

Accessing Class Constants

- **Format** (outside of the class definition)[1]:

```
<Class name>.<constant name>;
```

- **Example** (outside of the class definition):

```
main()
{
    System.out.println("Max life span: " + Person.MAX_AGE);
}
```

- Accessing a class constant inside the class where it's been defined does not require the name of the class

```
public class Person {
    public final static int MAX_AGE = 144;
    public void sayMax() { System.out.println(MAX_AGE); }
}
```

James Tam

Recap: Static Vs. Final

- **Static**: Means there's one instance of the attribute for the class (not individual instances for each instance (object) of the class)
- **Final**: Means that the attribute cannot change (it is a constant)

```
public class Foo
{
    public static final int num1= 1;
    private static int num2;  /*  Rare */
    public final int num3 = 1;  /* Why bother (waste) */
    private int num4;
     :           :
}
```

James Tam

An Example Class With A Static Implementation

```
public class Math
{
    // Public constants
    public static final double E = 2.71…
    public static final double PI = 3.14…

    // Public methods
    public static int abs(int a);
    public static long abs(long a);
      ...    ...    ...
}
```

- For more information about this class go to:
  - http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

James Tam

Should A Class Be Entirely Static?

- Usually purely static classes (cannot be instantiated) have only methods and no data (maybe some constants).
  - Rare: mostly cases there's variable data that is different from object-to-object so few classes are purely static
- Example (purely for illustration):

```
Math math1 = new Math();
Math math2 = new Math();
// What's the difference? Why bother?
math1.abs() vs. math2.abs();
```

- When in doubt *DO NOT* make attributes and methods static.

James Tam

After This Section You Should Now Know

- Static attributes and methods
  - How to create statics
  - How to access statics
  - When something should be static vs. non-static (instance)
  - How to represent static in UML
- How to declare class constants
  - The difference between static and final

James Tam

Copyright Notification

- “Unless otherwise indicated, all images in this presentation are used with permission from Microsoft.”