Static Keyword

Farzana Rahman

What is static

- The static keyword is used when a member variable of a class has to be shared between all the instances of the class.
- All static variables and methods belong to the class and not to any instance of the class

When can we access static variable

- When a class is loaded by the virtual machine all the static variables and methods are available for use.
- Hence we don' t need to create any instance of the class for using the static variables or methods.
- Variables which don' t have static keyword in the definition are implicitly non static.

Example

```
Class staticDemo{
      public static int a = 100; // All instances of staticDemo have this variable as a common `
                                            variable
      public int b =2 ;
      public static showA(){
                System.out.println("A is "+a);
      }
}
Class execClass{
      public static void main(String args[]){
                staticDemo.a = 35; // when we use the class name, the class is loaded, direct access to
                                            a without any instance
                staticDemo.b=22; // ERROR this is not valid for non static variable
                staticDemo demo = new staticDemo();
                demo.b = 200;  // valid to set a value for a non static variable after creating an instance.
                staticDemo.showA();  //prints 35
      }
}
```

Static and Non-static

- We can access static variables without creating an instance of the class
- As they are already available at class loading time, we can use them in any of our non static methods.
- We cannot use non static methods and variables without creating an instance of the class as they are bound to the instance of the class.
- They are initialized by the constructor when we create the object using new operator.

Why do we need this

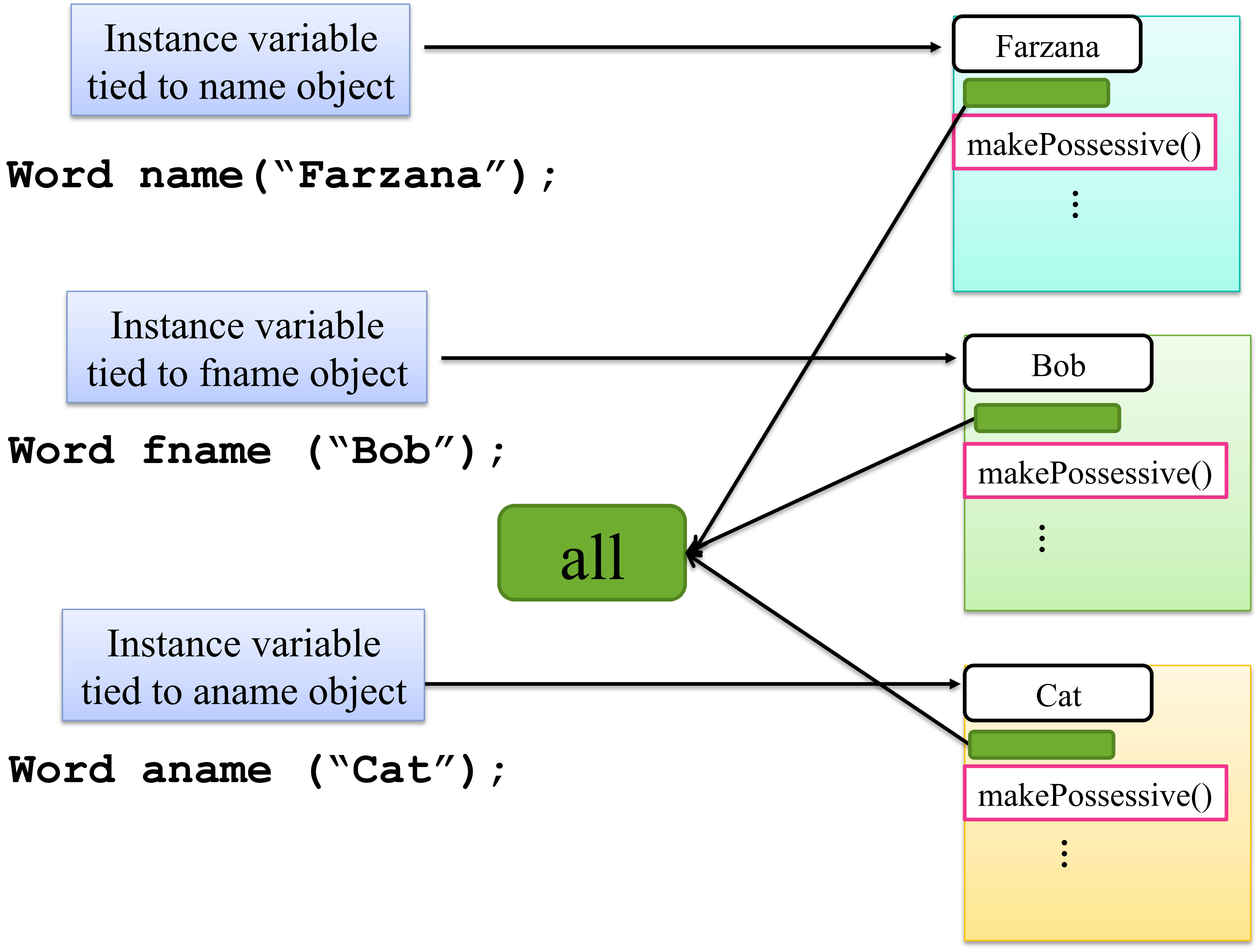
- Static methods are identified to be mostly used when we are writing any utility methods.
- We can also use static variables when sharing data.
- When sharing data do keep in mind about multithreading can cause inconsistency in the value. (synchronize the variable)

```
public class Word
{
  private String  inword;
  public static String all;

   public Word(String str)
  {
     inword = str;
     all= all+ " " + str;
  }
…
}
```

Different objects will have different instance variables

The entire class shares this

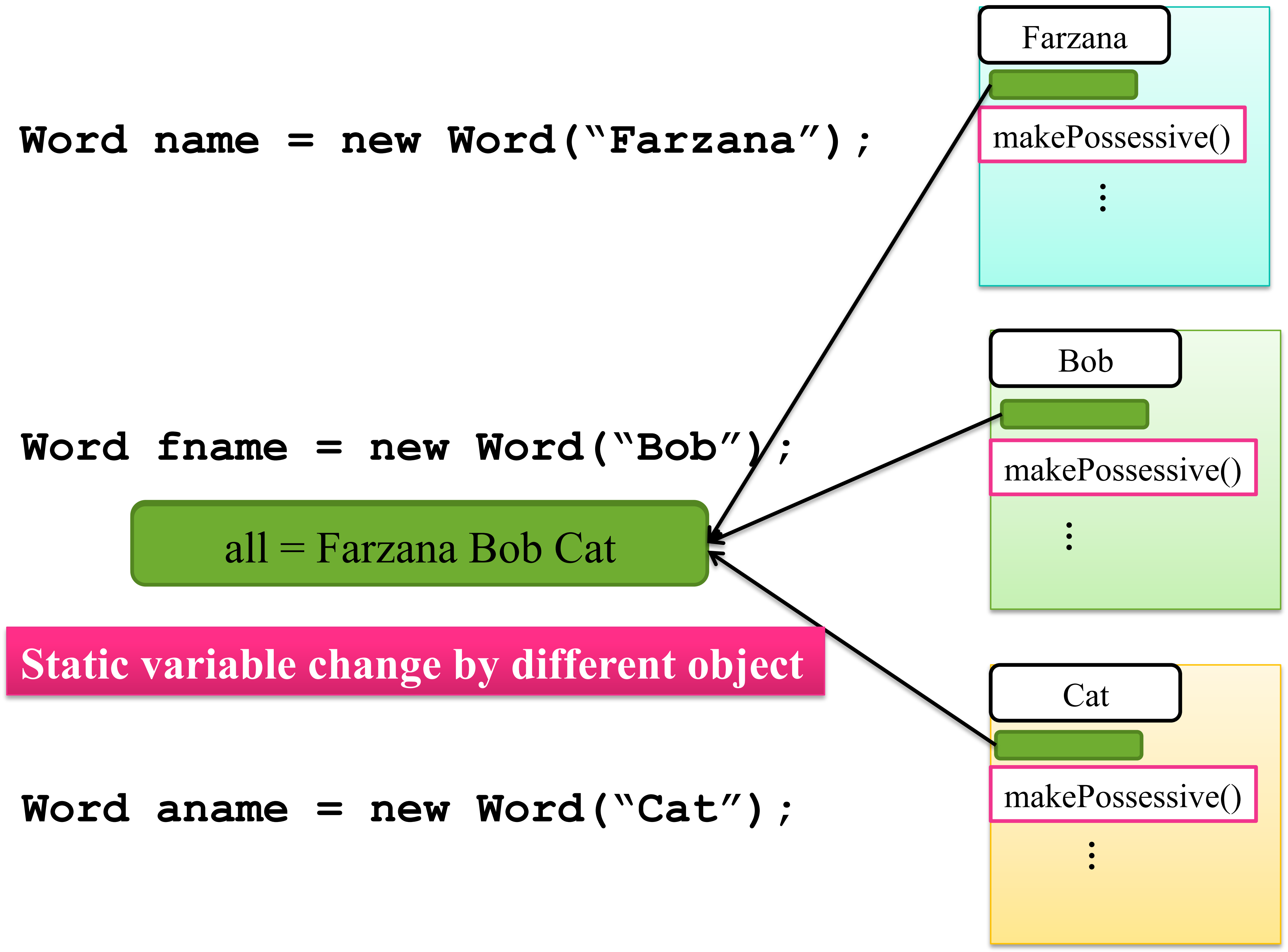


```
Word name("Farzana");

Word fname ("Bob");

Word aname ("Cat");
```

Instance variable tied to name object
Instance variable tied to fname object
Instance variable tied to aname object
Farzana
makePossessive()
Bob
makePossessive()
Cat
makePossessive()
all



```
Word name = new Word("Farzana");

Word fname = new Word("Bob");

Word aname = new Word("Cat");
```

Static variable change by different object