Class methods, class attributes and the class method main

Advanced Programming

Class attributes

- Class attributes are attributes that refer to the class as a whole and are shared among all the objects of the class
- The Java jargon refers to them with the term "static attributes"
- Class attributes cannot be stored in an object, as they are shared among all objects that are instances of the same class
  - Java class attributes are stored in an area of the heap specifically devoted to them → **"area for statics"**

Class attributes

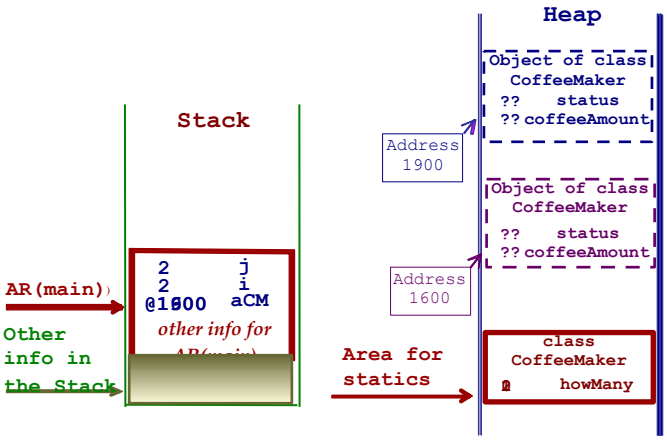
- Extent: from the beginning of a computation to its end
- Scope: the same as the scope of the class they belong
- In Java, declared with the keyword static

Class attributes

- Their initialization is performed as for other variables, with the direct initialization in the line of definition
  - public static int howMany = 0;
- This initialization is done only once
  - If it is initialized inside a constructor then the initialization may be repeated
  - A typical use of class attributes is to count the number of objects of a given class

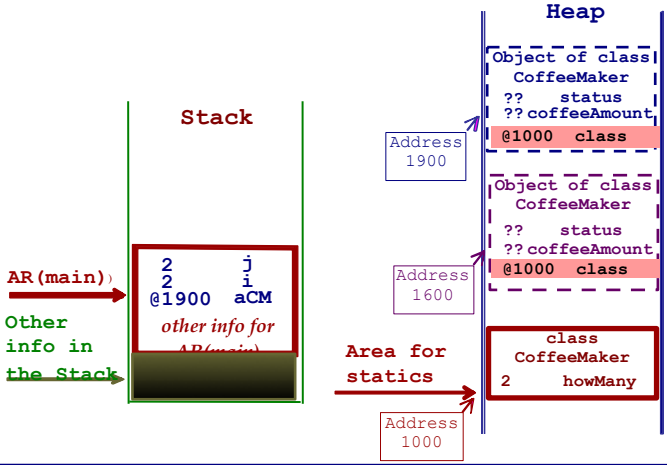
Class attributes in Java (1/2)

```
public class CoffeeMaker {
...
    public int coffeeAmount;
    public int status;
    public static int howMany = 0;
    public CoffeeMaker() {
       howMany++;
    }
    ...
    public static void main(String argv[]){
      CoffeeMaker aCM;
      aCM = new CoffeeMaker();
      aCM = new CoffeeMaker();
      int i = aCM.howMany;
      int j = CoffeeMaker.howMany;
    }
    ...
}
```



Class attributes in Java (2/2)

- The object of a class must know where the class variables for its class are located. Thus, each object, when created, obtains an, automatically created and to the correct value set, reference to its class



Accessing class attributes

- Via an object instance of the class
  - As if they were object attributes
  - `int i = aCM.howMany;`
- Via the class directly
  - It is used the same operator used to access the object attributes "."
  - `int j = CoffeeMaker.howMany;`

Class methods (1/2)

- Methods that can be invoked by a class
- Using the keyword `static` these methods are associated directly with the class
- Less effort: Static method is used in most cases when there is stand alone utility method that should be made available without requiring the overhead of instantiation

Class methods (2/2)

- A class method does not need objects to be invoked
- A class methods cannot access object attributes, that is, attributes that refer to a specific object instance of the class
- Its behavior is the same regardless of the object through it is invoked or whether it is invoked via the class name
- A class method does not require a reference to "`this`" in its activation record, as there may not exist any such objects

When to use class methods

- If you are writing utility classes and they are not supposed to be changed.
- If the method is not using any instance variable.
- If any operation is not dependent on instance creation.
- If there is some code that can easily be shared by all the instance methods, extract that code into a static method.

When to use class methods

- If you are sure that the definition of the method will never be changed or overridden. As static methods can not be overridden.
- A good candidate for a static method is a method that only works with the arguments provided to it
- Careful that with static methods you cannot override.

The class method main

- The execution of a program identifies a target class and asks the JVM to execute a class method on the target class, the main method
- Command line statement prompt> java CoffeeMaker
  - The operating system runs the JVM with parameter CoffeeMaker
  - The JVM sends a message to the class CoffeeMaker to invoke the class method main

Handling arguments

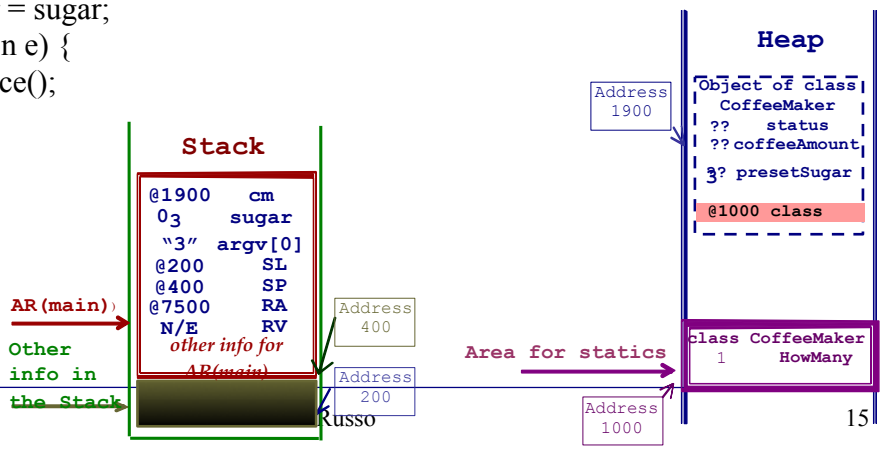
- Main method has as parameter an array of Strings
- If we want to create from command line an object of class CoffeeMaker with a default value of sugar equal to 3,

prompt> java CoffeeMaker 3

- The main method of CoffeeMaker is instantiated with an array of size 1 as its parameter, whose first element, argv[0], is equal to the string "3"

The class method `main`

```
public static void main(String argv[]) {
  int sugar = 0;
  CoffeeMaker cm = new CoffeeMaker();
  if (argv.length == 1) {
        String sugarString = argv[0];
        try {
          sugar = Integer.parseInt(sugarString);
          cm.presetSugar = sugar;
        } catch (Exception e) {
          e.printStackTrace();
        }
  }
 ...
}
```



Example

```
[1] public class C {
[2]   static int g = 0;
[3]   String clname;
[4]   public C(){
[5]      g++;
[6]   }
[7]   public C (String c){
[8]      this();
[9]      this.clname = c;
[10]  }
[11]  public static int shownumber(){
[12]     return g;
[13]  }
[14]  public static void main (String [] args){
[15]     int i;
[16]     C myC = new C();
[17]     C my1C = new C("Class2");
[18]     i = C.shownumber();
[19]  }
[20]}
```


Draw the stack diagram

- What is the value of i?
- Draw the stack diagram for when the execution reaches:
  - The end of line [16]
  - The end of line [17]
  - The end of line [18]
- Note:
  - SAR (global) @300

Source Code

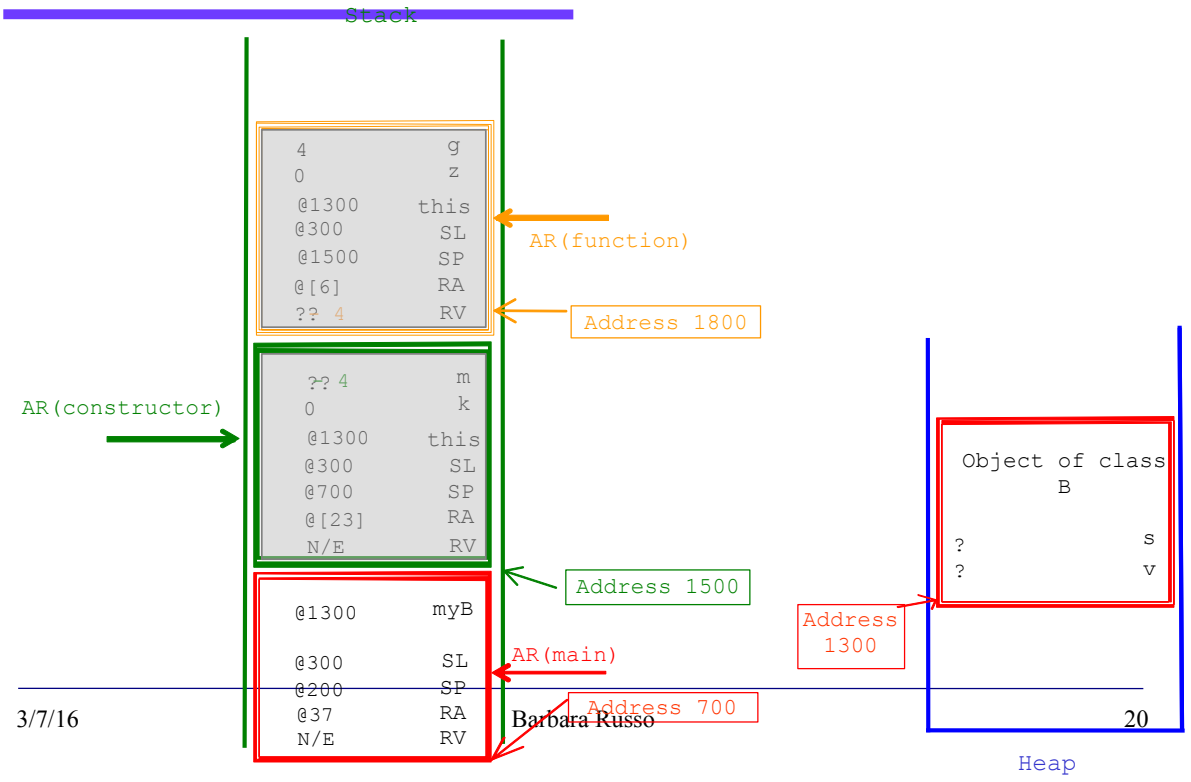
```
[1] public class B {
[2]   int v;
[3]   int s;
[4]   public B(){
[5]       int k = 0;
[6]       int m = function(k);
[7]   }
[8]
[9]   public B (int z){
[10]        int s = 34;
[11]  }
[12]  public B (double d){
[13]        s = (int) d;
[14]        v = (int) d+6;
[15]  }
[16]
[17]  private int function(int z){
[18]      int g = 4;
[19]      return  g+z;
[20]  }
```

```
[21]   public static void main (…){
[22]       B myB = new B();
[23]       B my2B = new B(3);
[24]       B my3B = new B(1.0);
[25]       my3B = myB;
[26]       int s = myB.function(12);
[27]       myB.v = s;
[28]   }
[29]}
```

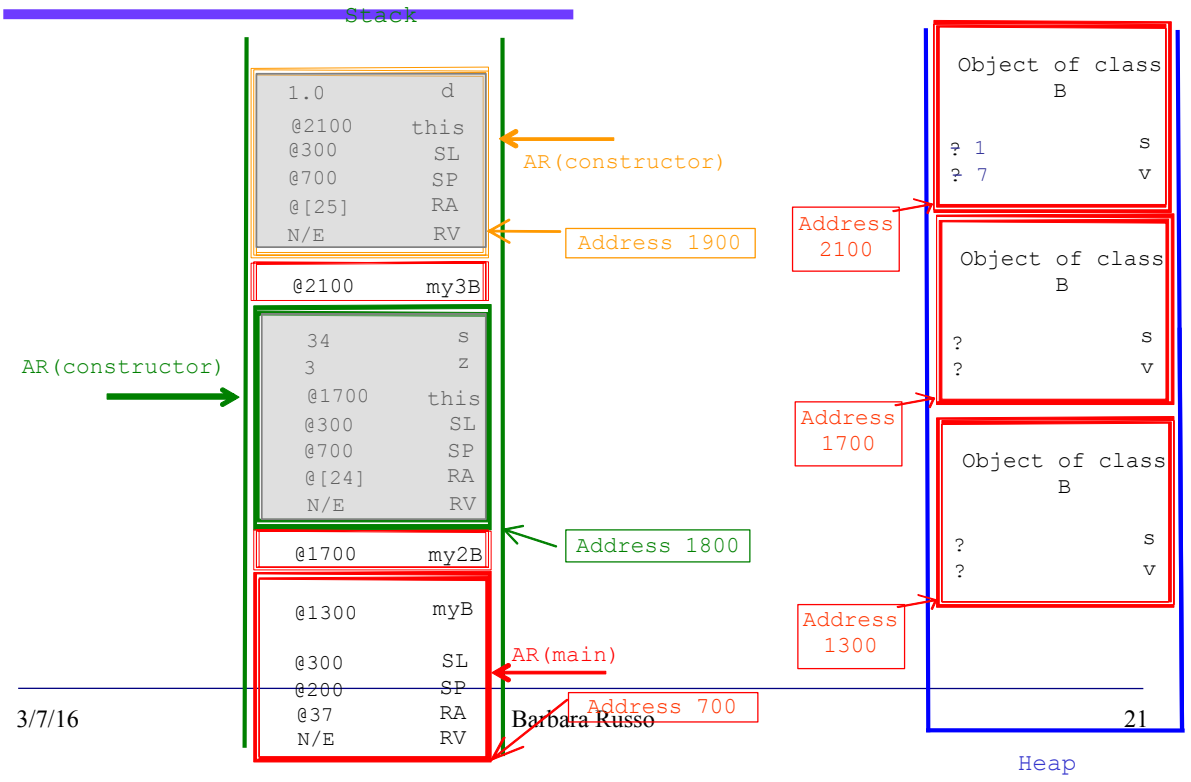

Draw the stack diagram

- Draw the stack diagram for when the execution reaches:
  - The end of line [22]
  - The end of line [24]
  - The end of line [27]
- Note:
  - SAR (global)à @300

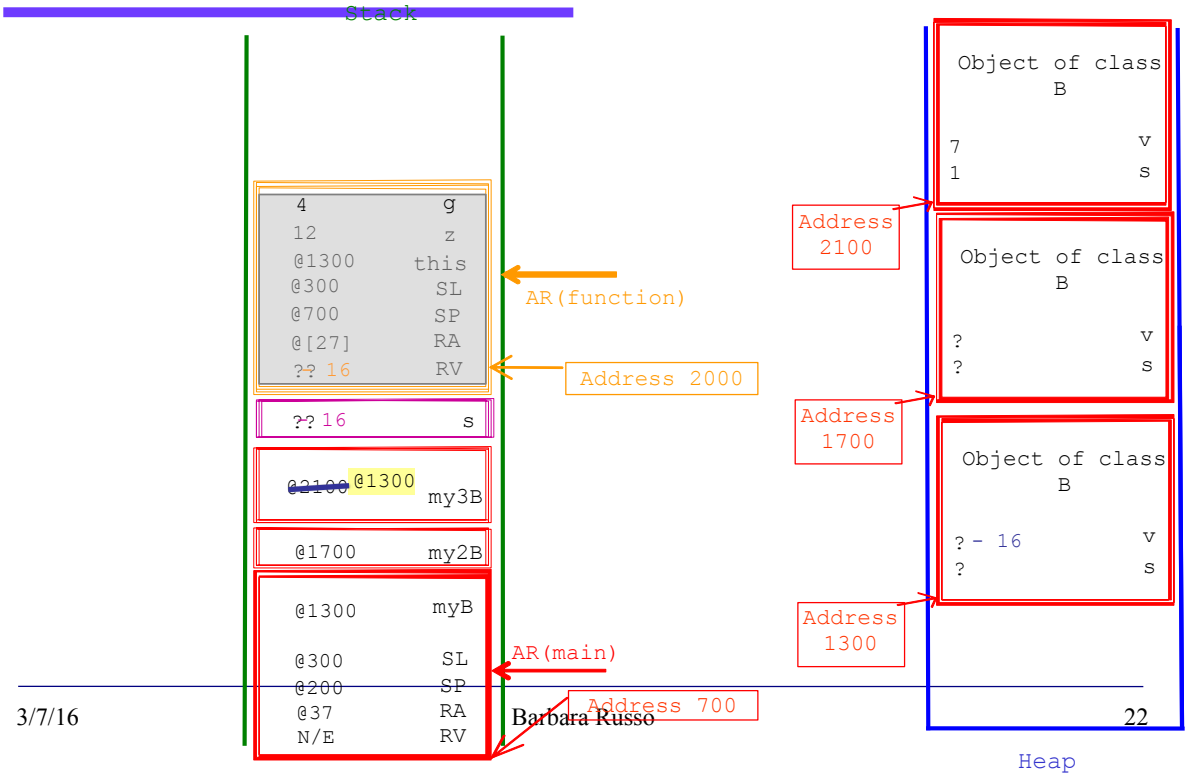
Solution 1



Solution 2



Solution 3


Stack
4 g
12 z
@1300 this
@300 SL
@700 SP
@[27] RA
?? 16 RV
?? 16 s
@2100 @1300 my3B
@1700 my2B
@1300 myB
@300 SL
@200 SP
@37 RA
N/E RV
AR(function)
Address 2000
AR(main)
Address 700
Object of class B
7 v
1 s
Address 2100
Object of class B
? v
? s
Address 1700
Object of class B
? - 16 v
? s
Address 1300
Heap