Static variables and methods

Introduction to OO Programming

`static` variables

- Java variables can be preceded by the `static` keyword.

- To declare a `static` int variable

```
private static int nextAccNo;
```

What is a `static` variable

- Only one copy of a `static` class variable exists for all objects of a class
- If a class contains a `static` variable, no matter how many objects of that class are created, they will all share the same `static` variable
- A `static` variable is also known as a `class` variable.

What is a `static` variable

- The value of the `static` variable is the same for all the class objects. If any one of the objects of the class modifies the value of a `static` variable, then the value of that `static` variable changes for every object of the class. (*only one copy*)

- Static variables are always underlined in a UML class diagram.

- Static variables are useful in keeping track of account numbers and for storing data that applies to all instances of a class – for example an interest rate that applies to all loans.

```
public class BankAccount
   {
      private int accountNo;
      private double balance;
      private static int nextAccNo = 1;

      //constructor
      public BankAccount()
      {
         accountNo = nextAccNo++;
         balance = 0;
      }

      //other code here.....

   }
```

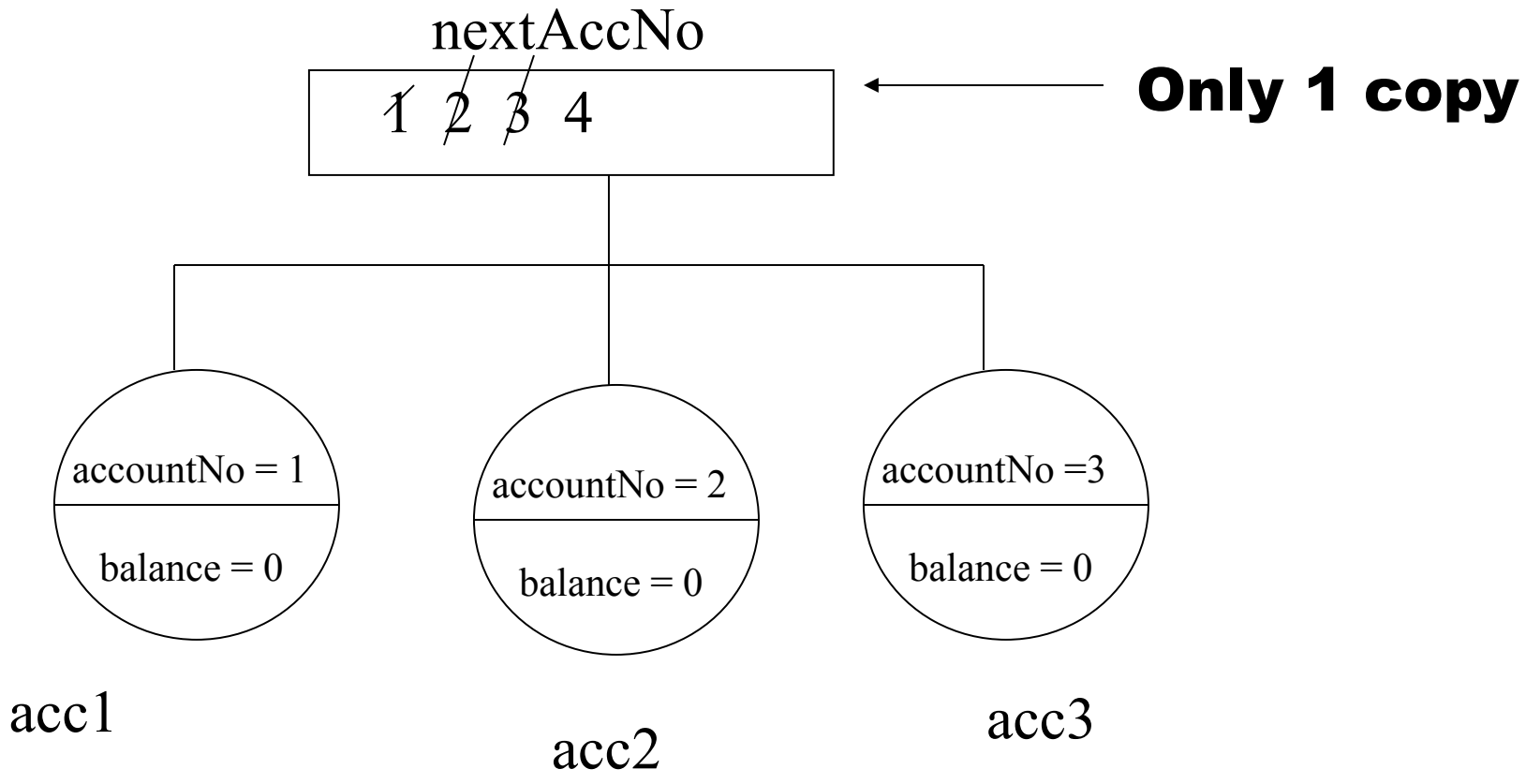
Example

- `BankAccount` has two instance variables `accountNo` and `balance`.
- It also has a `static` variable – `nextAccNo`. This `static` variable will be shared by all objects of the class `BankAccount`.
- In contrast, copies of the variables `accountNo` and `balance` are created every time a `BankAccount` object is created.

```
public class BankAccountTester
{
   public static void main(String args[])
   {

   BankAccount acc1 = new BankAccount();
   BankAccount acc2 = new BankAccount();
   BankAccount acc3 = new BankAccount();

   System.out.print("acc1 no:" +acc1.getAccountNo());
   System.out.print("acc2 no:" +acc2.getAccountNo());
   System.out.print("acc3 no:" +acc3.getAccountNo());


   }


}
```

Example

- In `main`, 3 BankAccount objects are created, `acc1`, `acc2` and `acc3`.
- When the first object (acc1) is created the `static` value of `nextAccNo` is 1 and the constructor is called.
- The `accountNo` field is initialized to the static value of `nextAccNo`(1) before `nextAccNo` is incremented and becomes 2
- The `balance` instance variable is also initialized to 0.


nextAccNo
1 2 3 4
Only 1 copy
accountNo = 1
balance = 0
accountNo = 2
balance = 0
accountNo =3
balance = 0
acc1
acc2
acc3

Assigning a value to static variable

- Because the `static` variable is independent of objects of the class it is assigned a value only once – when the class is loaded.

- Therefore a `static` variable is assigned a value as it is declared. It is not assigned a value in the constructor

```
private static int nextAccNo = 1;
```

- This value can subsequently be changed by methods or constructors

`static` methods

- Every method must be in a class
- A `static` method is a method which is not invoked on a specific object. An object does not need to be created for a `static` method to be called.
- A `static` method is also known as a `class` method

`static` methods

- Any method in Java, whose behaviour does not depend on the value of an instance variable, can be made `static`.
- The java classes `Math` and `Character` contain `static` methods
- Methods act on the arguments passed to them and do not depend on instance variables of class
- `Math` has method called `round()`, which takes a floating point argument and rounds it to the nearest integer.

```
System.out.print(Math.round(3.56));
```

How it works

- The constructor for `Math` class is private, preventing objects of the class being created

```
//will not work
Math mathObj = new Math();
int x = mathObj.round(12.4543);
```

- A `static` method, is a method of a class that does not use any instance variables of the class.

Calling `static` methods

- To call a `static` method such as `round()`, we can simply give the name of the class, followed by the dot (.) operator, followed by the method call.

- Because the method does not use any instance variables there is no need to declare an object of the class in order to call the methods of the class.

```
System.out.print(Math.round(23.4567));
double d = Math.sqrt(111);
```

Calling methods

- To call a `static` method you do not need an instance of the class, you just use the *class name*.

```
System.out.print(Math.round(3.56));
```

- To call a non-`static` method you do need an instance of the class.

```
String myName = new String("Hello");
System.out.print(myName.charAt(0));
```

`static` method definition

- A `static` method can be defined in any class.

```
public static void setInterestRate(double rateIn)
{
    interestRate = rateIn;
}
```

- Again, a static method is called without declaring an instance of the class.

```
BankAccount.setInterestRate(4.99);
```

The `final` keyword

- The java keyword `final`, can be used with variables, methods and classes.

- A `final` variable means you cannot change its value – it is a **constant**.

```
private final double PI = 3.14159;
private final int FREE_TRANSACTIONS = 3;
```

- You can also have `final` methods and classes. These will be covered in second year programming