STATIC MEMBERS AND METHODS

Static Members

- Java supports definition of global methods and variables that can be accessed without creating objects of a class. Such members are called Static members.
- Define a variable by marking with the static keyword .
- This feature is useful when we want to create a variable common to all instances of a class.
- One of the most common example is to have a variable that could keep a count of how many objects of a class have been created.
- Note: Java creates only one copy for a static variable which can be used even if the class is never instantiated.

Static Variables

- Define using *static:*

```
public class Circle {
     // class variable, one for the Circle class, how many circles
    public static int numCircles;

     //instance variables,one for each instance of a Circle
    public double x,y,r;
    // Constructors...
}
```

- Access with the class name (ClassName.StatVarName):

```
nCircles = Circle.numCircles;
```

Static Variables - Example

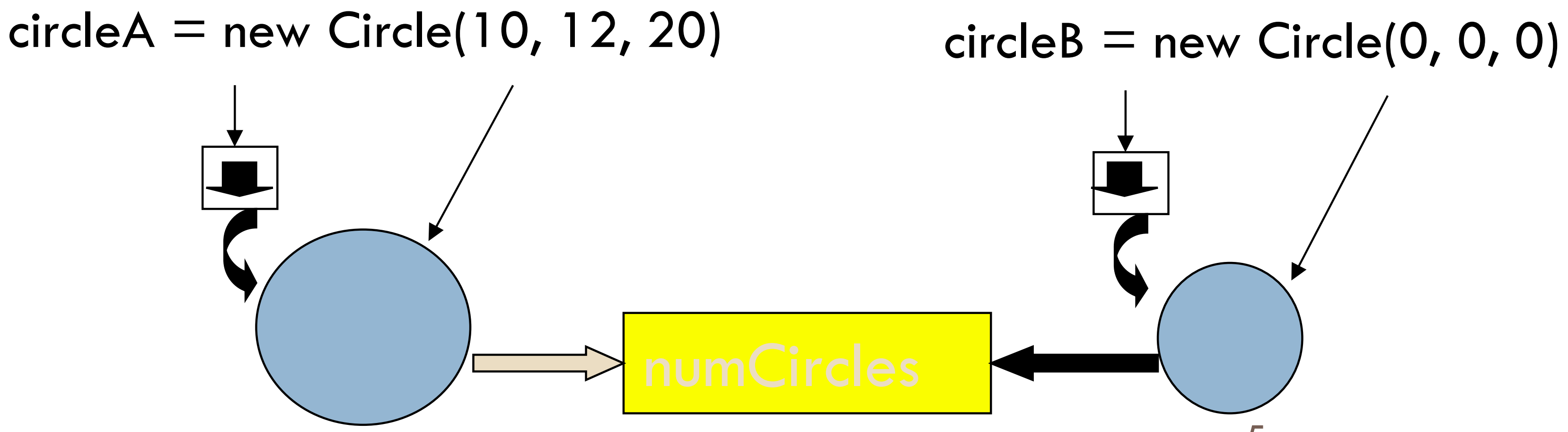
- Using *static variables:*

```
public class Circle {
     // class variable, one for the Circle class, how many circles
    public static int numCircles = 0;
    private double x,y,r;

    // Constructors...
Circle (){
        numCircles++; }
    Circle (double x, double y, double r){
        this.x = x;
        this.y = y;
        this.r = r;
        numCircles++;     }
CalculateCircumference()
Get and Set for all..
}
```

Class Variables - Example

- Using *static variables:*

```
public class CircleRunner{

        public static void main(String args[]){
            Circle circleA = new Circle( 10, 12, 20);    // numCircles = 1
            Circle circleB = new Circle();               // numCircles = 2
            S.O.P ( Circle.numCircle);
    }
}
```


circleA = new Circle(10, 12, 20)
circleB = new Circle(0, 0, 0)
numCircles

Non-static Vs Static Variables

- Non-static variables : One copy per object. Every object has its own instance variable.
  - E.g. x, y, r (centre and radius in the circle)

- Static variables : One copy per class.
  - E.g. numCircles (total number of circle objects created)

Important Points

- *Use a static variable when all objects of a class must use the same copy of the variable.*
- Static variables have class scope. We can access a class's public static members through a reference to any object of the class, or by qualifying the member name with the class name and a dot (.)
- A class's private static class members can be accessed by client code only through methods of the class.

Static Methods

- A class can have methods that are defined as static (e.g., main method).
- Static methods can be accessed without using objects. Also, there is NO need to create objects.
- They are prefixed with keyword “static”
- Static methods are generally used to group related library functions that don’t depend on data members of its class. For example, Math library functions.

The **`Math`** Class

- The **`Math`** class provides a number of standard mathematical methods
  - It is found in the **`java.lang`** package, so it does not require an **`import`** statement
  - All of its methods and data are static, therefore they are invoked with the class name **`Math`** instead of a calling object
  - The **`Math`** class has two predefined constants, **`E`** (*e*, the base of the natural logarithm system) and **`PI`** (π, 3.1415 . . .)

```
area = Math.PI * radius * radius;
```

Some Methods in the Class **`Math`** (Part 1 of 5)

Display 5.6 **Some Methods in the Class `Math`**

The `Math` class is in the `java.lang` package, so it requires no `import` statement.

```
public static double pow(double base, double exponent)
```

Returns `base` to the power `exponent`.

EXAMPLE

`Math.pow(2.0,3.0)` returns `8.0`.

(continued)

Some Methods in the Class **Math** (Part 2 of 5)

ison

Display 5.6 Some Methods in the Class Math

```
public static double abs(double argument)
public static float abs(float argument)
public static long abs(long argument)
public static int abs(int argument)
```

Returns the absolute value of the argument. (The method name abs is overloaded to produce four similar methods.)

EXAMPLE

Math.abs(−6) and Math.abs(6) both return 6. Math.abs(−5.5) and Math.abs(5.5) both return 5.5.

```
public static double min(double n1, double n2)
public static float min(float n1, float n2)
public static long min(long n1, long n2)
public static int min(int n1, int n2)
```

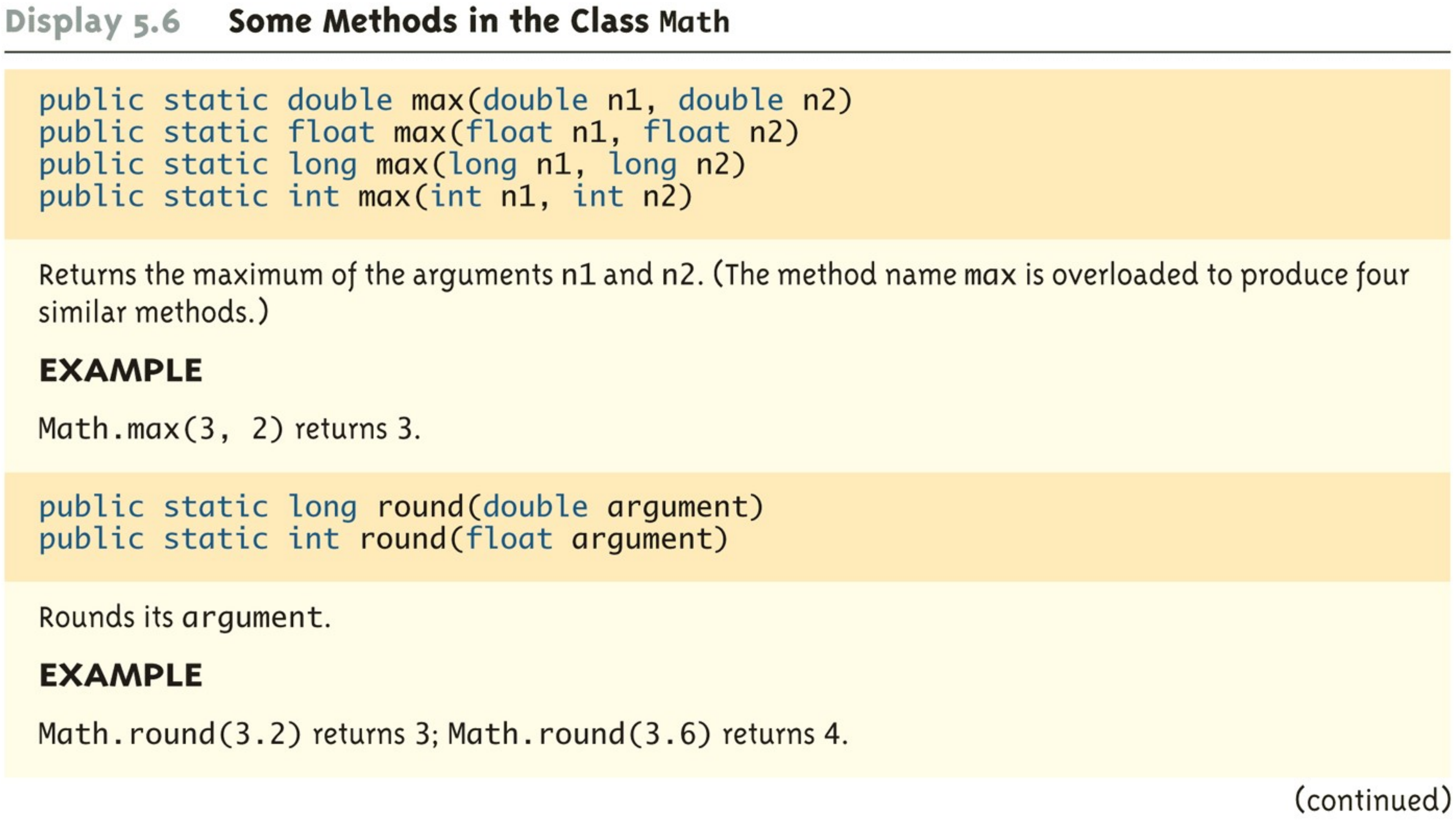
Returns the minimum of the arguments n1 and n2. (The method name min is overloaded to produce four similar methods.)

EXAMPLE

Math.min(3, 2) returns 2.

(continued)

Some Methods in the Class **`Math`** (Part 3 of 5)



Display 5.6 **Some Methods in the Class `Math`**

```
public static double max(double n1, double n2)
public static float max(float n1, float n2)
public static long max(long n1, long n2)
public static int max(int n1, int n2)
```

Returns the maximum of the arguments `n1` and `n2`. (The method name `max` is overloaded to produce four similar methods.)

EXAMPLE

`Math.max(3, 2)` returns 3.

```
public static long round(double argument)
public static int round(float argument)
```

Rounds its `argument`.

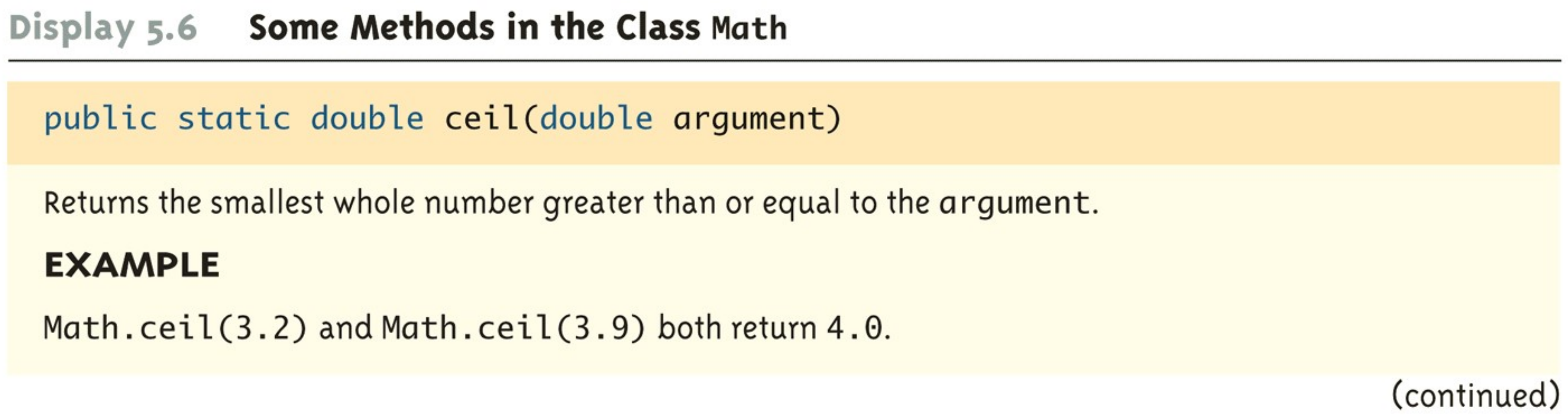
EXAMPLE

`Math.round(3.2)` returns 3; `Math.round(3.6)` returns 4.

(continued)

Some Methods in the Class **Math** (Part 4 of 5)

ison



Display 5.6 **Some Methods in the Class Math**

```
public static double ceil(double argument)
```

Returns the smallest whole number greater than or equal to the `argument`.

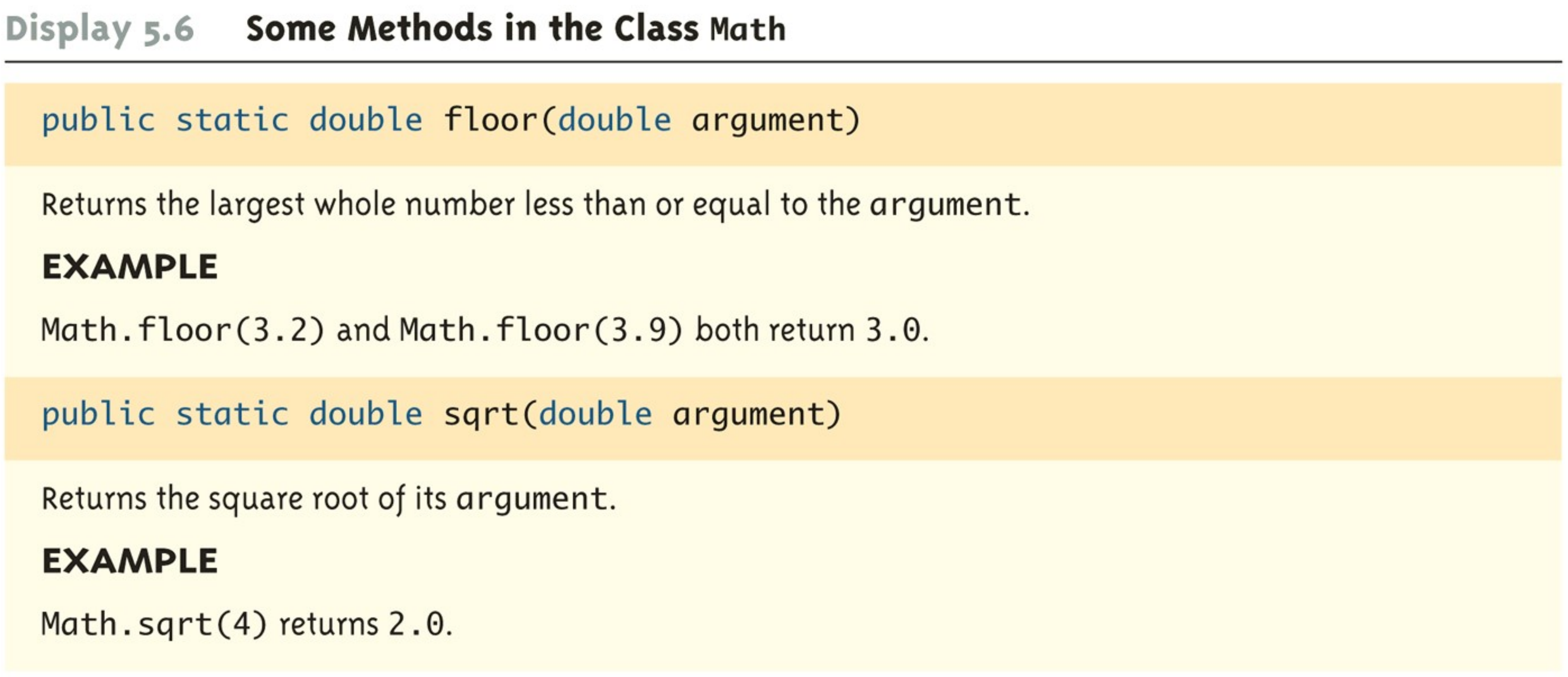
EXAMPLE

`Math.ceil(3.2)` and `Math.ceil(3.9)` both return `4.0`.

(continued)

Some Methods in the Class **`Math`** (Part 5 of 5)

ison



Display 5.6 **Some Methods in the Class `Math`**

```
public static double floor(double argument)
```

Returns the largest whole number less than or equal to the `argument`.

EXAMPLE

`Math.floor(3.2)` and `Math.floor(3.9)` both return `3.0`.

```
public static double sqrt(double argument)
```

Returns the square root of its `argument`.

EXAMPLE

`Math.sqrt(4)` returns `2.0`.

Comparator class with Static methods

```
// Comparator.java: A class with static data items comparision methods
class Comparator {
        public static int max(int a, int b)
        {
                if( a > b)
                        return a;
                else
                        return b;
        }

        public static String max(String a, String b)
        {
                if( a.compareTo (b) > 0)
                        return a;
                else
                        return b;
        }
}
```

Directly accessed using ClassName (NO Objects)

```
class MyClass {
  public static void main(String args[])
  {
        String s1 = "Melbourne";
        String s2 = "Sydney";
        String s3  = "Adelaide";

        int a = 10;
        int b = 20;

        System.out.println(Comparator.max(a, b)); // which number is big
        System.out.println(Comparator.max(s1, s2)); // which city is big
        System.out.println(Comparator.max(s1, s3)); // which city is big
  }
}
```

Static methods restrictions

- They can only call other static methods.
- They can only access static data.
- They cannot refer to “this” or “super” (more later) in anyway.

- A static method cannot access non-static class members, because a static method can be called even when no objects of the class have been instantiated.
- For the same reason, the this reference cannot be used in a static method. The this reference must refer to a specific object of the class, and when a static method is called, there might not be any objects of its class in memory.

Example – static data and methods

- Create a SavingsAccount class.
- Use a static data member annualInterestRate to store the annual interest rate.
- The class contains a private data member savingsBalance indicating the balance of account.
- Provide member function calculateMonthlyInterest that calculates the monthly interest by multiplying the balance by annualInterestRate divided by 12; this interest should be added to savingsBalance.
- Provide a static member function modifyInterestRate that sets the static annualInterestRate to a new value.

- Write a driver program to test class SavingsAccount. Instantiate two different objects of class SavingsAccount, saver1 and saver2, with balances of $2000.00 and $3000.00, respectively.
- Set the annualInterestRate to 3 percent.
- Then calculate the monthly interest and print the new balances for each of the savers.
- Then set the annualInterestRate to 4 percent, calculate the next month's interest and print the new balances for each of the savers.

Example - Calculator

- Create a class TwoDigitCalculator which allows user to perform addition, subtraction, multiplication and division on 2 digits.

- Analyse the data members and methods of this class and implement

END