Methods

Instructor: Mainak Chaudhuri
mainakc@cse.iitk.ac.in

Methods in problem solving

- Often we think of solving a large problem in parts
  - Computing the number of digits in n! involves two major steps: computing k=n! and computing the number of digits in k
  - So I can have two "procedures", one computes n! and the other computes the number of digits in it
  - Such procedures are called methods
    - These are just like functions
  - A method may or may not produce a value
  - The type of the produced value determines the "return type" of a method

Methods in problem solving

- Methods are useful in carrying out similar type of computation repeatedly
  - Imagine evaluating a polynomial $P(x, n) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ for a particular pair (x_0, n_0)
  - It is enough to have a single method that takes a, x and n as arguments or parameters and computes the value of ax^n
  - We repeatedly "call" this method to evaluate $P(x_0, n_0)$
  - Methods improve "modularity" of your program

Return and parameter type

- Can be int, float, String, double, char, or void
- We have seen one method so far: main
- It does not produce any value: void

```
public static void PrintMyName () {
    System.out.println("Tintin");
}
```

- Every method must be a part of a class

PrintMyName

```
class anExample {
    public static void main (String arg[]) {
        PrintMyName();     // Method call
    }

    public static void PrintMyName () {
        // Method declaration
        System.out.println("Tintin");
    }
}
```

Method parameters

- Methods can have parameters also
  - Just like functions

```
class anExample {
    public static void main (String arg[]) {
        String myName = "Tintin";
        PrintMyName(myName);
    }

    public static void PrintMyName (String s) {
        System.out.println(s);
    }
}
```

Method parameters

```
class moreExample {
    public static void main (String arg[]) {
        int n = 5;
        double x = 4.6;
        ComputeExponent(x, n);
    }

    public static void ComputeExponent (double
    base, int index) {
        double y = 1.0;
        System.out.println("Base: " + base + ", index:
    " + index);
        // continued in next slide
```

Method parameters

```
        if (index < 0) {
            base = 1/base;
            index = -index;
        }
        while (index > 0) {
            y *= base;
            index--;
         }
        System.out.println("Required answer: " + y);
    }
}
```

Producing values and using them

- Methods can return values to the calling method
  - The calling method can use these values to compute further
- Suppose we want to check if $2^x + 2^{2x+1} + 1$ is a perfect square
  - Involves two major procedures: computing $2^x + 2^{2x+1} + 1$ and checking if it is a square

Example of method

```
class exampleOfReturnValue {
    public static void main (String arg[]) {
        int x = 4, y = 1, z;  // assume x is positive
        boolean isSquare;
        z = ComputePowerOfTwo(x);
        y += z;
        z = ComputePowerOfTwo(2*x+1); // Reuse
        y += z;
        isSquare = CheckSquare(y);
        if (isSquare) {
            System.out.println("Expression is square
    for x = " + x);
        }
    }
```

Example of method

```
public static int ComputePowerOfTwo
(int index) {
    int y = 1;  // This is a different y
    while (index > 0) {
        y *= 2;
        index--;
     }
     return y;
}
```

Example of method

```
public static boolean CheckSquare (int x)
{
    int y;        // This is another y
    for (y=0; y*y <= x; y++) {
        if (y*y == x) {
            return true;
        }
    }
    return false;
}
} // end class
```

More examples

```
class Trigonometry {
   public static void main (String arg[]) {
      double error, x = 0.785;  // quarter pi
      error = cos(x)*cos(x) + sin(x)*sin(x) – 1;
      System.out.println("Error: " + error);
   }
   // continued in next slide
```

More examples

```
public static double cos (double theta) {
   return (1 – theta*theta/2 +
ComputeExponent(theta, 4)/factorial(4));
}

public static double sin (double theta) {
    return (theta –
ComputeExponent(theta, 3)/factorial(3)
+ ComputeExponent(theta, 5)/
factorial(5));
}
```

More examples

```
public static double ComputeExponent (double
a, int n) {
    double y = 1.0;
    if (n < 0) {
        a = 1/a;
        n = -n;
    }
    while (n > 0) {
        y *= a;
        n--;
     }
     return y;
}
```

More examples

```
    public static int factorial (int n) {
        int y = 1;
        while (n > 1) {
            y *= n;
            n--;
        }
        return y;
    }
} // end class Trigonometry
```

Checking character case

```
class characterCase {
   public static void main  (String arg[]) {
      char c = 'A';
      if (isUpperCase(c)) {
          System.out.println(c + " is upper case.");
      }
      else if (isLowerCase(c)) {
           System.out.println(c + " is lower case.");
      }
      else {
          System.out.println(c + " is not in English
  alphabet.");
      }
   }// continued in next slide
```

Checking character case

```
public static boolean isUpperCase (char
c) {
   return (isBetween('A', c, 'Z'));
}

public static boolean isLowerCase (char
c) {
    return (isBetween('a', c, 'z'));
}
// continued in next slide
```

Checking character case

```
    public static boolean isBetween (char a,
    char b, char c) {
        return ((a <= b) && (b <= c));
    }
} // end class
```

Case conversion

```
class convertCase {
    public static void main (String arg[]) {
        char c = 'M';
        System.out.println("Original: " + c);
        if (isBetween('A', c, 'Z')) {
            System.out.println("After conversion: " +
    toLower(c));
        }
        else if (isBetween('a', c, 'z')) {
            System.out.println("After conversion: " +
    toUpper(c));
        }
        else {
            System.out.println("Cannot convert case!");
        }
    } // continued in the next slide
```

Case conversion

```
    public static boolean isBetween (char a, char b,
   char c) {
        return ((a <= b) && (b <= c));
    }

    public static char toLower (char c) {
        return (c – 'A' + 'a');
    }

    public static char toUpper (char c) {
        return (c – 'a' + 'A');
    }
} // end class
```