**Redundant recipes**

- **Recipe for baking 20 cookies:**
  - Mix the following ingredients in a bowl:
    * 4 cups flour
    * 1 cup butter
    * 1 cup sugar
    * 2 eggs
    * 40 pounds chocolate chips ...
  - Place on sheet and Bake for about 10 minutes.

- **Recipe for baking 40 cookies:**
  - Mix the following ingredients in a bowl:
    * 8 cups flour
    * 2 cups butter
    * 2 cups sugar
    * 4 eggs
    * 80 pounds chocolate chips ...
  - Place on sheet and Bake for about 10 minutes.

---

**Parameterized recipe**

- **Recipe for baking 20 cookies:**
  - Mix the following ingredients in a bowl:
    * 4 cups flour
    * 1 cup sugar
    * 2 eggs
    * ...

- **Recipe for baking N cookies:**
  - Mix the following ingredients in a bowl:
    * N/5 cups flour
    * N/20 cups butter
    * N/20 cups sugar
    * N/10 eggs
    * 2N bags chocolate chips ...
  - Place on sheet and Bake for about 10 minutes.

  - **parameter:** A value that distinguishes similar tasks.

---

**Redundant figures**

- **Consider the task of printing the following lines/boxes:**
  
  *************
  
  *******
  
  ********************
  
  *************
  
  * *
  
  *******
  
  * *
  
  *******
A redundant solution

public class Stars1 {
    public static void main(String[] args) {
        lineOf13();
        lineOf7();
        lineOf35();
        box10x3();
        box5x4();
    }

    public static void lineOf13() {
        for (int i = 1; i <= 13; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    public static void lineOf7() {
        for (int i = 1; i <= 7; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    public static void lineOf35() {
        for (int i = 1; i <= 35; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    ...
}

Parameterization

- This code is redundant.
- Would variables help?
- Would constants help?
- What is a better solution?
  - line - A method to draw a line of any number of stars.
  - box - A method to draw a box of any size.

Declaring a parameter

Stating that a method requires a parameter in order to run

public static void name (type name) {
    statement(s);
}

- Example:
  public static void sayPassword(int code) {
      System.out.println("The password is: " + code);
  }

  - When sayPassword is called, the caller must specify the integer code to print.

Passing a parameter

Calling a method and specifying values for its parameters

name (expression);

- Example:
  public static void main(String[] args) {
      sayPassword(42);
      sayPassword(12345);
  }

  Output:
  The password is 42
  The password is 12345
Parameters and loops

• A parameter can guide the number of repetitions of a loop.

```java
public static void main(String[] args) {
    chant(3);
}

public static void chant(int times) {
    for (int i = 1; i <= times; i++) {
        System.out.println("Just a salad...");
    }
}
```

Output:
Just a salad...
Just a salad...
Just a salad...

How parameters are passed

• When the method is called:
  – The value is stored into the parameter variable.
  – The method’s code executes using that value.

```java
public static void main(String[] args) {
    chant(3);
    chant(7);
}
```

```java
public static void chant(int times) {
    for (int i = 1; i <= times; i++) {
        System.out.println("Just a salad...");
    }
}
```

Common errors

• If a method accepts a parameter, it is illegal to call it without passing any value for that parameter.
  ```java
  chant(); // ERROR: parameter value required
  ```

• The value passed to a method must be of the correct type.
  ```java
  chant(3.7); // ERROR: must be of type int
  ```

• Exercise: Change the Stars program to use a parameterized method for drawing lines of stars.

Stars solution

```java
// Prints several lines of stars.
// Uses a parameterized method to remove redundancy.
public class Stars2 {
    public static void main(String[] args) {
        line(13);
        line(7);
        line(35);
    }

    // Prints the given number of stars plus a line break.
    public static void line(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print("*");
        }
        System.out.println();
    }
}
```
Multiple parameters

- A method can accept multiple parameters. (separate by ,)
  - When calling it, you must pass values for each parameter.

- Declaration:
  
  ```java
  public static void name (type name, ..., type name) {
      statement(s);
  }
  ```

- Call:
  
  ```java
  methodName(value, value, ..., value);
  ```

Multiple params example

```java
public static void main(String[] args) {
    printNumber(4, 9);
    printNumber(17, 6);
    printNumber(8, 0);
    printNumber(0, 8);
}
```

```java
public static void printNumber(int number, int count) {
    for (int i = 1; i <= count; i++) {
        System.out.print(number);
    }
    System.out.println();
}
```

Output:

```
444444444
171717171717
00000000
```

- Modify the Stars program to draw boxes with parameters.

Stars solution

```java
// Prints several lines and boxes made of stars.
// Third version with multiple parameterized methods.
public class Stars3 {
    public static void main(String[] args) {
        line(13);
        line(7);
        line(35);
        System.out.println();
        box(10, 3);
        box(5, 4);
        box(20, 7);
    }

    // Prints the given number of stars plus a line break.
    public static void line(int count) {
        for (int i = 1; i <= count; i++) {
            System.out.print("*");
        }
        System.out.println();
    }

    // Prints a box of stars of the given size.
    public static void box(int width, int height) {
        line(width);
        for (int line = 1; line <= height - 2; line++) {
            System.out.print(" ");
            for (int space = 1; space <= width - 2; space++) {
                System.out.print(" ");
            }
            System.out.print("*");
            System.out.println();
        }
        line(width);
    }
}
```
### Value semantics

- **value semantics**: When primitive variables (`int, double`) are passed as parameters, their values are copied.
  - Modifying the parameter will not affect the variable passed in.

```java
public static void strange(int x) {
    x = x + 1;
    System.out.println("1. x = " + x);
}
public static void main(String[] args) {
    int x = 23;
    strange(x);
    System.out.println("2. x = " + x);
}
```

Output:
```
1. x = 24
2. x = 23
```

### "Parameter Mystery" problem

```java
public class ParameterMystery {
    public static void main(String[] args) {
        int x = 9;
        int y = 2;
        int z = 5;
        mystery(z, y, x);
        mystery(y, x, z);
    }
    public static void mystery(int x, int z, int y) {
        System.out.println(z + " and " + (y - x));
    }
}
```

### Strings

- **string**: A sequence of text characters.
  ```java
  String name = "text";
  String name = expression;
  - Examples:
    ```java
    String name = "Marla Singer";
    int x = 3;
    int y = 5;
    String point = "(" + x + ", " + y + ")";
  ```

### Strings as parameters

```java
public class StringParameters {
    public static void main(String[] args) {
        sayHello("Marty");
        String teacher = "Bictolia";
        sayHello(teacher);
    }
    public static void sayHello(String name) {
        System.out.println("Welcome, " + name);
    }
}
```

Output:
```
Welcome, Marty
Welcome, Bictolia
```

- **Modify the Stars program to use string parameters. Use a method named repeat that prints a string many times.**
Stars solution

// Prints several lines and boxes made of stars.
// Fourth version with String parameters.
public class Stars4 {
    public static void main(String[] args) {
        line(13);
        line(7);
        line(35);
        System.out.println();
        box(10, 3);
        box(5, 4);
        box(20, 7);
    }

    // Prints the given number of stars plus a line break.
    public static void line(int count) {
        repeat("*", count);
        System.out.println();
    }

    // Prints a box of stars of the given size.
    public static void box(int width, int height) {
        line(width);
        for (int line = 1; line <= height - 2; line++) {
            System.out.print(" ");
            repeat(" ", width - 2);
            System.out.println("*");
        }
        line(width);
    }

    // Prints the given String the given number of times.
    public static void repeat(String s, int times) {
        for (int i = 1; i <= times; i++) {
            System.out.print(s);
        }
    }
}

Java's Math class

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math.abs(value)</td>
<td>absolute value</td>
</tr>
<tr>
<td>Math.ceil(value)</td>
<td>rounds up</td>
</tr>
<tr>
<td>Math.floor(value)</td>
<td>rounds down</td>
</tr>
<tr>
<td>Math.log10(value)</td>
<td>logarithm, base 10</td>
</tr>
<tr>
<td>Math.max(value1, value2)</td>
<td>larger of two values</td>
</tr>
<tr>
<td>Math.min(value1, value2)</td>
<td>smaller of two values</td>
</tr>
<tr>
<td>Math.pow(base, exp)</td>
<td>base to the exp power</td>
</tr>
<tr>
<td>Math.random()</td>
<td>random double between 0 and 1</td>
</tr>
<tr>
<td>Math.round(value)</td>
<td>nearest whole number</td>
</tr>
<tr>
<td>Math.sqrt(value)</td>
<td>square root</td>
</tr>
<tr>
<td>Math.sin(value)</td>
<td>sine/cosine/tangent of an angle in radians</td>
</tr>
<tr>
<td>Math.cos(value)</td>
<td></td>
</tr>
<tr>
<td>Math.tan(value)</td>
<td></td>
</tr>
<tr>
<td>Math.toDegrees(value)</td>
<td>convert degrees to radians and back</td>
</tr>
<tr>
<td>Math.toRadians(value)</td>
<td></td>
</tr>
</tbody>
</table>

Constant Description
- Math.E: 2.7182818...
- Math.PI: 3.1415926...
## Calling Math methods

Math.\texttt{name}(\texttt{parameters})

- **Examples:**
  
  ```java
  double squareRoot = Math.sqrt(121.0); // 11.0
  System.out.println(squareRoot);
  int absoluteValue = Math.abs(-50); // 50
  System.out.println(absoluteValue);
  System.out.println(Math.min(3, 7) + 2); // 5
  ```

- **The Math methods do not print to the console.**
  - Each method produces ("returns") a numeric result.
  - The results are used as expressions (printed, stored, etc.).

## Return

- **return:** To send out a value as the result of a method.
  - The opposite of a parameter:
    - Parameters send information \textit{in} from the caller to the method.
    - Return values send information \textit{out} from a method to its caller.
      - A call to the method can be used as part of an expression.

## Math questions

- **Evaluate the following expressions:**
  - Math.abs(-1.23)
  - Math.pow(3, 2)
  - Math.pow(10, -2)
  - Math.sqrt(121.0) - Math.sqrt(256.0)
  - Math.ceil(6.022) + Math.floor(15.9994)
  - Math.abs(Math.min(-3, -5))

- **Some Math methods return double or other non-int types.**
  ```java
  int x = Math.pow(10, 3); // ERROR: incompat. types
  ```

- **Some double values print poorly (too many digits).**
  ```java
  double result = 1.0 / 3.0;
  System.out.println(result); // 0.3333333333333333
  ```

- **The computer represents doubles in an imprecise way.**
  ```java
  System.out.println(0.1 + 0.2);
  ```
  - Instead of 0.3, the output is 0.3000000000000004

## Quirks of real numbers

- Some Math methods return double or other non-int types.

- Some double values print poorly (too many digits).

- The computer represents doubles in an imprecise way.
### Type casting

- **type cast**: A conversion from one type to another.
  - To promote an `int` into a `double` to get exact division from `/`
  - To truncate a `double` from a real number to an integer

- Syntax:

  `(type) expression`

**Examples:**

```
double result = (double) 19 / 5;  // 3.8
int result2 = (int) result;        // 3
int x = (int) Math.pow(10, 3);    // 1000
```

### More about type casting

- Type casting has high precedence and only casts the item immediately next to it.

```
- double x = (double) 1 + 1 / 2;   // 1
- double y = 1 + (double) 1 / 2;   // 1.5
```

- You can use parentheses to force evaluation order.

```
- double average = (double) (a + b + c) / 3;
```

- A conversion to `double` can be achieved in other ways.

```
- double average = 1.0 * (a + b + c) / 3;
```

### Returning a value

```
public static type name(parameters) {
    statements;
    ...
    return expression;
}
```

**Example:**

```
// Returns the slope of the line between the given points.
public static double slope(int x1, int y1, int x2, int y2) {
    double dy = y2 - y1;
    double dx = x2 - x1;
    return dy / dx;
}
```

```
- slope(1, 3, 5, 11) returns 2.0
```

### Return examples

- **Converts degrees Fahrenheit to Celsius.**

```
public static double fToC(double degreesF) {
    double degreesC = 5.0 / 9.0 * (degreesF - 32);
    return degreesC;
}
```

- **Computes triangle hypotenuse length given its side lengths.**

```
public static double hypotenuse(int a, int b) {
    double c = Math.sqrt(a * a + b * b);
    return c;
}
```

**You can shorten the examples by returning an expression:**

```
public static double fToC(double degreesF) {
    return 5.0 / 9.0 * (degreesF - 32);
}
```
Common error: Not storing

- Many students incorrectly think that a return statement sends a variable's name back to the calling method.

```java
public static void main(String[] args) {
    slope(0, 0, 6, 3);
    System.out.println("The slope is " + result); // ERROR: // result not defined
}

public static double slope(int x1, int x2, int y1, int y2) {
    double dy = y2 - y1;
    double dx = x2 - x1;
    double result = dy / dx;
    return result;
}
```

Fixing the common error

- Instead, returning sends the variable's value back.
  - The returned value must be stored into a variable or used in an expression to be useful to the caller.

```java
public static void main(String[] args) {
    double s = slope(0, 0, 6, 3);
    System.out.println("The slope is " + s);
}

public static double slope(int x1, int x2, int y1, int y2) {
    double dy = y2 - y1;
    double dx = x2 - x1;
    double result = dy / dx;
    return result;
}
```

Classes and objects

- **class**: A program entity that represents either:
  1. A program / module, or
  2. A type of objects.
  - A class is a blueprint or template for constructing objects.
  - Example: The `DrawingPanel` class (type) is a template for creating many `DrawingPanel` objects (windows).
    - Java has 1000s of classes. Later (Ch.8) we will write our own.

- **object**: An entity that combines data and behavior.
  - object-oriented programming (OOP): Programs that perform their behavior as interactions between objects.
**Objects**

- **object**: An entity that contains data and behavior.
  - **data**: variables inside the object
  - **behavior**: methods inside the object
    - You interact with the methods; the data is hidden in the object.

- Constructing (creating) an object:
  ```java
  Type objectName = new Type(parameters);
  ```

- Calling an object's method:
  ```java
  objectName.methodName(parameters);
  ```

**Blueprint analogy**

- iPod blueprint/factory
  - **state**: current song, volume, battery life
  - **behavior**: power on/off, change station/song, change volume, choose random song

- iPod #1
  - **state**: song = "1,000,000 Miles", volume = 17, battery life = 2.5 hrs
  - **behavior**: power on/off, change station/song, change volume, choose random song

- iPod #2
  - **state**: song = "Letting You", volume = 9, battery life = 3.41 hrs
  - **behavior**: power on/off, change station/song, change volume, choose random song

- iPod #3
  - **state**: song = "Discipline", volume = 24, battery life = 1.8 hrs
  - **behavior**: power on/off, change station/song, change volume, choose random song

**Strings**

- **string**: An object storing a sequence of text characters.
  - Unlike most other objects, a String is not created with `new`.
    ```java
    String name = "text";
    String name = expression;
    ```

  - Examples:
    ```java
    String name = "Marla Singer";
    int x = 3;
    int y = 5;
    String point = "(" + x + ", " + y + ")";
    ```

**Indexes**

- Characters of a string are numbered with 0-based **indexes**:
  ```java
  String name = "woodpile";
  ```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>w</td>
<td>o</td>
<td>o</td>
<td>d</td>
<td>p</td>
<td>i</td>
<td>l</td>
<td>e</td>
</tr>
</tbody>
</table>

  - First character's index: 0
  - Last character's index: 1 less than the string's length
  - The individual characters are values of type char (seen later)
### String methods

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>indexOf(str)</code></td>
<td>index where the start of the given string appears in this string (-1 if not found)</td>
</tr>
<tr>
<td><code>length()</code></td>
<td>number of characters in this string</td>
</tr>
<tr>
<td><code>substring(index1, index2)</code> or <code>substring(index1)</code></td>
<td>the characters in this string from <code>index1</code> (inclusive) to <code>index2</code> (exclusive); if <code>index2</code> is omitted, grabs till end of string</td>
</tr>
<tr>
<td><code>toLowerCase()</code></td>
<td>a new string with all lowercase letters</td>
</tr>
<tr>
<td><code>toUpperCase()</code></td>
<td>a new string with all uppercase letters</td>
</tr>
</tbody>
</table>

- These methods are called using the dot notation:

  ```java
  String gangsta = "Dr. Dre";
  System.out.println(gangsta.length());  // 7
  ```

### String method examples

```java
// index 012345678901
String s1 = "Stuart Reges";
String s2 = "Marty Stepp";
System.out.println(s1.length());  // 12
System.out.println(s1.indexOf("e"));  // 8
System.out.println(s1.substring(7, 10));  // "Reg"
String s3 = s2.substring(1, 7);
System.out.println(s3.toLowerCase());  // "arty s"
```

- Given the following string:

  ```java
  // index 012345678901234567890123456789012345678901
  String book = "Building Java Programs";
  ```

  - How would you extract the word "Java"?

### Modifying strings

- Methods like `substring` and `toLowerCase` build and return a new string, rather than modifying the current string.

  ```java
  String s = "lil bow wow";
  s.toUpperCase();
  System.out.println(s);  // LIL BOW WOW
  ```

- To modify a variable's value, you must reassign it:

  ```java
  String s = "lil bow wow";
  s = s.toUpperCase();
  System.out.println(s);  // LIL BOW WOW
  ```

### Interactive Programs with Scanner
**Input and System.in**

- **interactive program**: Reads input from the console.
  - While the program runs, it asks the user to type input.
  - The input typed by the user is stored in variables in the code.
  - Can be tricky; users are unpredictable and misbehave.
  - But interactive programs have more interesting behavior.

- **Scanner**: An object that can read input from many sources.
  - Communicates with `System.in` (the opposite of `System.out`)
  - Can also read from files (Ch. 6), web sites, databases, ...

**Scanner syntax**

- The **Scanner class** is found in the `java.util` package.

  ```java
  import java.util.*;    // so you can use Scanner
  ```

- Constructing a **Scanner object** to read console input:

  ```java
  Scanner name = new Scanner(System.in);
  ```

  - Example:
    ```java
    Scanner console = new Scanner(System.in);
    ```

**Scanner methods**

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nextInt()</code></td>
<td>reads an <code>int</code> from the user and returns it</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>reads a <code>double</code> from the user</td>
</tr>
<tr>
<td><code>next()</code></td>
<td>reads a one-word <code>String</code> from the user</td>
</tr>
<tr>
<td><code>nextLine()</code></td>
<td>reads a one-line <code>String</code> from the user</td>
</tr>
</tbody>
</table>

- Each method waits until the user presses Enter.
- The value typed by the user is returned.

  ```java
  System.out.print("How old are you? ");    // prompt
  int age = console.nextInt();
  System.out.println("You typed " + age);
  ```

- **prompt**: A message telling the user what input to type.

**Scanner example**

```java
import java.util.*;    // so that I can use Scanner
public class UserInputExample {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("How old are you? ");    // prompt
        int age = console.nextInt();
        System.out.println("You typed " + age);
        int years = 65 - age;
        System.out.println(years + " years to retirement!");
    }
}
```

- Console (user input underlined):

  How old are you? 29
  36 years until retirement!
import java.util.*; // so that I can use Scanner
public class ScannerMultiply {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Please type two numbers: ");
        int num1 = console.nextInt();
        int num2 = console.nextInt();
        int product = num1 * num2;
        System.out.println("The product is "+ product);
    }
}

- Output (user input underlined):
  Please type two numbers: 8 6
  The product is 48
  - The Scanner can read multiple values from one line.

Strings as user input

- Scanner’s `next` method reads a word of input as a String.
  Scanner console = new Scanner(System.in);
  System.out.print("What is your name? ");
  String name = console.next();
  name = name.toUpperCase();
  System.out.println(name + " has " + name.length() + " letters and starts with " + name.substring(0, 1));

  Output:
  What is your name? Chamillionaire
  CHAMILLIONAIRE has 14 letters and starts with C

- The `nextLine` method reads a line of input as a String.
  System.out.print("What is your address? ");
  String address = console.nextLine();

Strings question

- Write a program that outputs a person’s "gangsta name."
  - first initial
  - Diddy
  - last name (all caps)
  - first name
  - izzle
  
  Example Output:
  Type your name, playa: Marge Simpson
  Your gangsta name is "M. Diddy SIMPSON Marge-izzle"

Input tokens

- **token**: A unit of user input, as read by the Scanner.
  - Tokens are separated by *whitespace* (spaces, tabs, new lines).
  - How many tokens appear on the following line of input?
    23  John Smith  42.0  "Hello world"  $2.50  "  19"

- When a token is not the type you ask for, it crashes.
  System.out.print("What is your age? ");
  int age = console.nextInt();

  Output:
  What is your age? Timmy
  java.util.InputMismatchException
  at java.util.Scanner.nextInt(Unknown Source)
  at java.util.Scanner.nextInt(Unknown Source)
  ...
// This program prints your "gangsta" name.
import java.util.*;

public class GangstaName {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);
        System.out.print("Type your name, playa: ");
        String name = console.nextLine();

        // split name into first/last name and initials
        String first = name.substring(0, name.indexOf(" "));
        String last = name.substring(name.indexOf(" ") + 1);
        last = last.toUpperCase();
        String fInitial = first.substring(0, 1);

        System.out.println("Your gangsta name is " + fInitial + ". Diddy " + last + " " + first + "-izzle");
    }
}