Procedural decomposition using static methods
Recall: An algorithm is a list of steps for solving a problem.

What is the algorithm to bake sugar cookies?
The “Bake sugar cookies” algorithm

- Mix the dry ingredients.
- Cream the butter and sugar.
- Beat in the eggs.
- Stir in the dry ingredients.
- Set the oven for the appropriate temperature.
- Set the timer.
- Place the cookies into the oven.
- Allow the cookies to bake.
- Mix the ingredients for the frosting.
- Spread frosting and sprinkles onto the cookies.
Structured algorithm

1. Make the cookie batter.
   - Mix the dry ingredients.
   - Cream the butter and sugar.
   - Beat in the eggs.
   - Stir in the dry ingredients.

2. Bake cookies.
   - Set the oven for the appropriate temperature.
   - Set the timer.
   - Place the cookies into the oven.
   - Allow the cookies to bake.

3. Add frosting and sprinkles.
   - Mix the ingredients for the frosting.
   - Spread frosting and sprinkles onto the cookies.

- **Observation:** Structured algorithms are easier to understand.
How do we bake a double batch?

Unstructured:
- Mix the dry ingredients.
- Cream the butter and sugar.
- Beat in the eggs.
- Stir in the dry ingredients.
- Set the oven …
- Set the timer.
- Place the cookies into the oven.
- Allow the cookies to bake.
- Set the oven …
- Set the timer.
- Place the cookies into the oven.
- Allow the cookies to bake.
- Mix the ingredients for the frosting.
- Spread frosting and sprinkles onto the cookies.

Structured:
- 1. Make the cookie batter.
- 2a. Bake the first batch of cookies.
- 2b. Bake the second batch of cookies.
- 3. Add frosting and sprinkles.

Observation: Structured algorithms eliminate redundancy.
Redundancy in programs

```java
public class FraggleRock {
    public static void main(String[] args) {
        System.out.println("Dance your cares away,");
        System.out.println("Worry's for another day.");
        System.out.println("Let the music play,");
        System.out.println("Down at Fraggle Rock.");
    }
}
```
How do we structure the program?

- **static method**: A group of statements given a name.

- To use a static method:
  1. **define** it (write down the recipe)
     - Write a group of statements and give it a name.
  2. **call** it (cook using the recipe)
     - Tell our program to execute the method.
public class FraggleRock {

    public static void singChorus() {
        System.out.println("Dance your cares away, ");
        System.out.println("Worry's for another day.");
        System.out.println("Let the music play,");
        System.out.println("Down at Fraggle Rock.");
    }

    public static void main(String[] args) {
        singChorus();
        System.out.println();
        singChorus();
    }
}
Declaring a static method

- The syntax for declaring a static method (writing down the recipe):

```java
public class <class name> {
    ...
    public static void <method name>() {
        <statement>;
        <statement>;
        <statement>;
        ...
        <statement>;
    }
}
```
Calling a static method

- The syntax for calling a static method (cooking using the recipe):

  `<method name>()` ;
Static method example

- Declaring a static method
  ```java
  public static void printAffirmation() {
      System.out.println("I am good enough!");
      System.out.println("I am smart enough!");
      System.out.println("People like me!");
  }
  ```

- Calling a static method (possibly multiple times)
  ```java
  printAffirmation();
  printAffirmation();
  ```

- Output
  ```
  I am good enough!
  I am smart enough!
  People like me!
  I am good enough!
  I am smart enough!
  People like me!
  ```
Methods calling methods

- One static method can call another:

```java
public class MethodsExample {
    public static void main(String[] args) {
        message1();
        message2();
        System.out.println("Done with main.");
    }

    public static void message1() {
        System.out.println("This is message1.");
    }

    public static void message2() {
        System.out.println("This is message2.");
        message1();
        System.out.println("Done with message2.");
    }
}
```

Output:

This is message1.
This is message2.
This is message1.
Done with message2.
Done with main.
Control flow of methods

- When a method is called, the execution
  - "jumps" into that method
  - executes all of the method’s statements
  - "jumps" back to the statement after the method call
public class MethodsExample {
    public static void main(String[] args) {
        message1();
        message2();
        ...
    }
}

public static void message1() {
    System.out.println("This is message1.");
}

public static void message2() {
    System.out.println("This is message2");
    message1();
    System.out.println("Done with message2.");
}

public static void message1() {
    System.out.println("This is message1.");
}
Summary: To use or not to use…

- **Yes**
  - Statements that are related to each other (*structure*).
  - Statements that are repeated (*redundancy*).

- **No**
  - Individual statements occurring only once and not related to other statements
  - Unrelated or weakly-related statements
    - Consider splitting the method into two smaller methods.
  - Blank lines
    - Blank `println` statements can go in the main method.
Example: Figure drawing

- Write a program to print the figures. Use static methods to capture structure and eliminate redundancy.
Version 1: Unstructured

- Create an empty program with a skeletal header and \texttt{main} method.

- Copy the expected output into it, surrounding each line with \texttt{System.out.println} syntax.

- Run and verify that it produces the correct output.
// Suzy Student, CSE 142, Autumn 2047
// This program prints several assorted figures.
//
public class Figures1 {
    public static void main(String[] args) {
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
        System.out.println("  ______");
        System.out.println(" /\");
        System.out.println("/\");
        System.out.println("| STOP |\");
        System.out.println("\____/\");
        System.out.println();
    }
}
Version 2: Structured with redundancy

- Identify the overall structure of the output, and divide the `main` method into several static methods based on this structure.
Version 2: Structured with redundancy

- Identify the overall structure of the output, and divide the main method into several static methods based on this structure.

The structure of the output:
- initial "egg" figure
- second "teacup" figure
- third "stop sign" figure
- fourth "hat" figure

This structure can be represented by methods:
- drawEgg
- drawTeaCup
- drawStopSign
- drawHat
Version 2: Structured with redundancy

// Suzy Student, CSE 142, Autumn 2047
// Prints several assorted figures, with methods for structure.

public class Figures2 {
    public static void main(String[] args) {
        drawEgg();
        drawTeaCup();
        drawStopSign();
        drawHat();
    }

    // Draws a figure that vaguely resembles an egg.
    public static void drawEgg() {
        System.out.println("  ______");
        System.out.println(" / \\
                        \\
                        /");
        System.out.println(" / \\
                        \\
                        /");
        System.out.println("/ \\
                        \\
                        \\
                        /");
        System.out.println();
    }

    // Draws a figure that vaguely resembles a teacup.
    public static void drawTeaCup() {
        System.out.println("/ \\
                        \\
                        \\
                        /");
        System.out.println("+--------+");
        System.out.println("\________/");
        System.out.println();
    }
}
// Draws a figure that vaguely resembles a stop sign.
public static void drawStopSign() {
    System.out.println("  ______");
    System.out.println(" / \ --- \ ");
    System.out.println("/ / | STOP |");
    System.out.println("| | \ // |");
    System.out.println("\ \_\_\_");
    System.out.println();
}

// Draws a figure that vaguely resembles a hat.
public static void drawHat() {
    System.out.println("  ______");
    System.out.println(" / \ --- \ ");
    System.out.println("/ / | |");
    System.out.println("| | \ // |");
    System.out.println("\ \_\_\_");
    System.out.println("+----------+");
}

Version 3: Structured without redundancy

- Further divide the program to eliminate all redundancy.
Version 3: Structured without redundancy

- Further divide the program to eliminate all redundancy.

The redundancy:
- top half of egg (purple)
- bottom half of egg (green)
- divider line (yellow)

This redundancy can be fixed by methods:
- drawEggTop
- drawEggBottom
- drawLine
Version 3: Structured without redundancy

// Suzy Student, CSE 142, Autumn 2047
// Prints several figures, with methods for structure and redundancy.

public class Figures3 {
    public static void main(String[] args) {
        drawEgg();
        drawTeaCup();
        drawStopSign();
        drawHat();
    }

    // draws redundant part that looks like the top of an egg
    public static void drawEggTop() {
        System.out.println("  ______");
        System.out.println(" / \  ");
        System.out.println("/ \ ");
    }

    // draws redundant part that looks like the bottom of an egg
    public static void drawEggBottom() {
        System.out.println("\ \_____/ ");
        System.out.println("\ \ / ");
        System.out.println(" \ \_____/ ");
    }
}
// Draws a figure that vaguely resembles an egg.
public static void drawEgg() {
    drawEggTop();
    drawEggBottom();
    System.out.println();
}

// Draws a figure that vaguely resembles a teacup.
public static void drawTeaCup() {
    drawEggBottom();
    System.out.println("+----------+");
    System.out.println();
}

// Draws a figure that vaguely resembles a stop sign.
public static void drawStopSign() {
    drawEggTop();
    System.out.println("| STOP |");
    drawEggBottom();
    System.out.println();
}

// Draws a figure that vaguely resembles a hat.
public static void drawHat() {
    drawEggTop();
    System.out.println("+----------+");
}
/* Suzy Student  
CS 101, Fall 2019  
This program prints lyrics from my favorite song! */
public class MyFavoriteSong {

/* Runs the overall program to print the song  
on the console. */
public static void main(String[] args) {
    sing();

    // Separate the two verses with a blank line  
    System.out.println();

    sing();
}

// Displays the first verse of the theme song.  
public static void sing() {
    System.out.println("Now this is the story all about how");
    System.out.println("My life got flipped turned upside-down");
}
}
Comments: How-to

- Do not describe the syntax/statements in detail.
- Instead, provide a short English description of the observed behavior when the method is run.

- Example:

```java
// This method prints the lyrics to the first verse
// of my favorite TV theme song.
// Blank lines separate the parts of the verse.
public static void verse1() {
    System.out.println("Now this is the story all about how");
    System.out.println("My life got flipped turned upside-down");
    System.out.println();
    System.out.println("And I'd like to take a minute,");
    System.out.println("just sit right there");
    System.out.println("I'll tell you how I became the prince");
    System.out.println("of a town called Bel-Air");
}
```
Exercise

- Write a program that prints the following output to the console. Use static methods as appropriate.
  I do not like my email spam,
  I do not like them, Sam I am!
  I do not like them on my screen,
  I do not like them to be seen.
  I do not like my email spam,
  I do not like them, Sam I am!

- Write a program that prints the following output to the console. Use static methods as appropriate.
  Lollipop, lollipop
  Oh, lolli lolli lolli

  Lollipop, lollipop
  Oh, lolli lolli lolli

  Call my baby lollipop
Exercise

- Write a program to print the block letters spelling "banana". Use static methods to capture structure and and eliminate redundancy.