Motivation for Methods

- Break up a complex problem into simpler sub-problems, which you can solve separately
  - E.g. Brining Turkey recipe and Stuffing Recipe
- Write once and reuse
  - DRY principle (“Don’t Repeat Yourself”)
- Methods are also known as procedures, subroutines, functions

About methods

- A method is a named group of statements
- You execute those statements by calling the method
- When you call the method, you can give it parameters (information)
- A method typically has a return value (a single piece of information coming out of the method)

Existing Methods for Boe-Bot

- CPU class provides already implemented methods for the controller to command Boe-Bot
  - Library – so you don’t re-invent the wheel
  - Need special statement to indicate where the code is located i.e. import stamp.core.CPU;
- Logic values or state for I/O pins
  - 1 – HIGH or TRUE
  - 0 – LOW or FALSE
- E.g. readPin(pin#) - reads state of the pin
  - writePin(pin#, logic value) – write to pin w/ 1/0
Method Syntax

```
return-type method-name(parameters)
{
    statements
}
```

- **Example 1:**
  ```java
  boolean isAdult(int age) {
      int magicAge = 21;
      return (age >= magicAge);
  }
  ```

- **Example 2:**
  ```java
  int average(int a, int b) {
      int c = (a + b) / 2;
      return c;
  }
  ```

---

Anatomy of a method

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Execute some action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Parts of your computation can be given a name. Usually a verb.</td>
</tr>
<tr>
<td></td>
<td>e.g. isAdult, controllingLED, turnLeft</td>
</tr>
<tr>
<td>Code</td>
<td>return-type method-name(parameters)</td>
</tr>
<tr>
<td>Output</td>
<td>Value of type as stated in return-type</td>
</tr>
<tr>
<td></td>
<td>e.g. return (age &gt;= magicAge);</td>
</tr>
<tr>
<td></td>
<td>If no output then return-type is void</td>
</tr>
<tr>
<td>Inputs</td>
<td>0 or more. Each input is associated with a type.</td>
</tr>
<tr>
<td>Usage</td>
<td>e.g. turn() or Wheels.turn() depending on where the method is accessed</td>
</tr>
</tbody>
</table>

---

Main Method

```java
public static void main() {...}
```

- **A special static method**
  - Whose return type is void
    - A void method does some work but does return anything

- **Entry point of a java program**
  - i.e. where the instructions starts to get executed step by step
    - If there is variable declared, then space is allocated in memory
    - If it comes across method call, then method declaration and statements are executed
    - Until the last statement, after which terminates the program
      - A void method can also end simply with "return;"

---

Keyword Static

- For now all methods must contain keyword static before the return-type of a method
  - E.g. static boolean isAdult(int age){ ... }

- Later we will see non-static methods with Object-Oriented Programming
  - For now all code must be within static methods
Option 1 w/ Main Method

```java
public class Age{
    public static boolean isAdult(int age) {
        int magicAge = 21;
        return (age >= magicAge);
    }
    public static void main() {
        boolean answer = isAdult(19) ;
        if(answer){
            System.out.println("Yes, Adult!");
        } else{
            System.out.println("No, Adult!");
        }
    }
}
```

Note: You can write any method in any order

Option 2 w/ Main Method

```java
//In TestAge.java
public class TestAge {
    public static void main() {
        boolean answer = Age.isAdult(19);
        if(answer){
            System.out.println("Yes, Adult!");
        } else{
            System.out.println("No, Adult!");
        }
    }
}
```

```java
//In Age.java
public class Age{
    public static boolean isAdult(int age) {
        int magicAge = 21;
        return (age >= magicAge);
    }
}
```

Return statements in loops

- Return statement should last statement before ending method
- Having return statements within loops will cause compiler to throw syntax error
  - This because the compiler does not know whether the statement is reachable or not
  - Always use variable to store the value and then finally return that value

Accessibility Level / Modifier

- To control the usage Methods (and Classes)
- `public:` makes the method accessible from outside the class
  - E.g. public static void main() { .. }
- `private:` not accessible outside the class
- In `default case:` (i.e. no mention of public and private): accessible if within same directory
- Accessibility level `appears before`
  - `static` keyword for static method
  - `class` keyword for class description