Methods

Problem

```
// (1) make sure the code only runs on mac os x
boolean mrjVersionExists = System.getProperty("mrj.version") != null;
boolean osNameExists = System.getProperty("os.name").startsWith("Mac OS");
if ( !mrjVersionExists || !osNameExists) {
    System.err.println("Not running on a Mac OS X system.");
    System.exit(1);
}

// (2) do all the logfile setup stuff
int currentLoggingLevel = DEFAULT_LOG_LEVEL;
File errorFile = new File(ERROR_LOG_FILENAME);
File warningFile = new File(WARNING_LOG_FILENAME);
File debugFile = new File(DEBUG_LOG_FILENAME);
// order of checks is important; want to go with more granular if multiple
if (errorFile.exists()) currentLoggingLevel = DDLoggerInterface.LOG_ERROR;
if (warningFile.exists()) currentLoggingLevel = DDLoggerInterface.LOG_WARN
if (debugFile.exists()) currentLoggingLevel = DDLoggerInterface.LOG_DEBUG;
logger = new DDSimpleLogger(CANON_DEBUG_FILENAME, currentLoggingLevel, tru
```

Solution

```
dieIfNotRunningOnMacOsX();
connectToLogfile();
connectToPreferences();
getDefaultColor();
```

Profit

- Human-readable
- Hide difficult code
- Don’t repeat yourself (**DRY**)

Methods

Methods: syntax

```
тип_возвращаемого_значения название_метода([параметры]) {
    // тело метода
}
```

Methods: example

```
public static void main(String[] args) {
    System.out.println("Hell to World!");
}
```

Methods: example

```
public class Program {
    public static void main(String args[]) {
        //TODO
    }

    void hello() {
        System.out.println("Hello");
    }

    void welcome() {
        System.out.println("Welcome to Java 11");
    }
}
```

Methods

```
имя_метода(аргументы);
```

Methods

```
public class Program {
    public static void main(String args[]) {
        hello();
        welcome();
        welcome();
    }

    static void hello() {
        System.out.println("Hello");
    }

    static void welcome() {
        System.out.println("Welcome to Java 11");
    }
}
```

Methods

```
Hello
Welcom to Java 11
Welcom to Java 11
```

Passing parameters to methods

Passing parameters to methods

```
static void sum(int x, int y) {
    int z = x + y;
    System.out.println(z);
}
```

Passing parameters to methods

```
public class Program {
    public static void main(String args[]) {
        int a = 6;
        int b = 8;
        sum(a, b); // 14
        sum(3, a); // 9
        sum(5, 23); // 28
    }

    static void sum(int x, int y) {
        int z = x + y;
        System.out.println(z);
    }
}
```

Passing parameters to methods

```
public class Program {
    public static void main(String args[]) {
        display("Tom", 34);
        display("Bob", 28);
        display("Sam", 23);
    }

    static void display(String name, int age) {
        System.out.println(name);
        System.out.println(age);
    }
}
```

Operator `return`

Operator `return`

```
public class Program {
    public static void main(String args[]) {
        int x = sum(1, 2, 3);
        int y = sum(1, 4, 9);
        System.out.println(x); // 6
        System.out.println(y); // 14
    }

    static int sum(int a, int b, int c) {
        return a + b + c;
    }
}
```

JMM (Java Memory Model)

```
public class Stack_Test {

 public static void main(String[] args) {

        int i=1;
        int j=2;

        Stack_Test reff = new Stack_Test();
        reff.foo(i);
    }

     void foo(int param) {
        int k = 3;
        System.out.println(param);
    }

}
```