
Design methods with parameters

COMP 102

Victoria University of Wellington

Designing with methods that call other methods

- Design a Java program to measure reaction time of users responding to true and false "facts".
 - Ask the user about a fact: e.g. "Is it true that the BE is a 4 Year degree?"
 - Measure the time they took
 - Print out how much time.
- Need a class
 - what name?
- Need a method
 - what name?
 - what parameters?
 - what actions?

ReactionTimeMeasurer

```
/** Measures reaction times for responding to true-false statements */
```

```
public class ReactionTimeMeasurer {
```

```
/** Measure and report the time taken to react to a question */
```

```
    public void measureReactionTime() {
```

```
        // find out the current time and remember it
```

```
        // ask the question and wait for answer
```

```
        // find out (and remember) the current time
```

```
        // print the difference between the two times
```

```
    }
```

```
}
```

Write the method body in comments first,
(to plan the method without worrying about syntax)

Work out what information needs to be stored (ie, variables)

ReactionTimeMeasurer

```
/** Measure and report the time taken to react to a question */
```

```
public void measureReactionTime() {
```

```
    long startTime = System.currentTimeMillis();
```

```
    UI.askString("Is it true that the sky is blue?");
```

```
    long endTime = System.currentTimeMillis();
```

```
    UI.printf("Reaction time = %d milliseconds \n", (endTime - startTime) );
```

```
}
```

```
}
```

Returns a very big integer
⇒ long
(milliseconds since 1/1/1970)

Just asking one question is not enough for an experiment.

→ need to ask a sequence of questions.

Multiple questions, the bad way

```

/** Measure and report the time taken to react to a question */
public void measureReactionTime(){
    long startTime = System.currentTimeMillis();
    UI.askString( "Is it true that John Quay was the Prime Minister");
    long endTime = System.currentTimeMillis();
    UI.printf("You took %d milliseconds \n", (endTime - startTime) );

    startTime = System.currentTimeMillis();
    UI.askString( "Is it true that 6 x 4 = 23");
    endTime = System.currentTimeMillis();
    UI.printf("You took %d milliseconds \n", (endTime - startTime) );

    startTime = System.currentTimeMillis();
    UI.askString( "Is it true that summer is warmer than winter");
    endTime = System.currentTimeMillis();
    UI.printf("You took %d milliseconds \n", (endTime - startTime) );

    startTime = System.currentTimeMillis();
    UI.askString( "Is it true that Wellington's population > 1,000,000");
    endTime = System.currentTimeMillis();
    UI.printf("You took %d milliseconds \n", (endTime - startTime) );
}

```

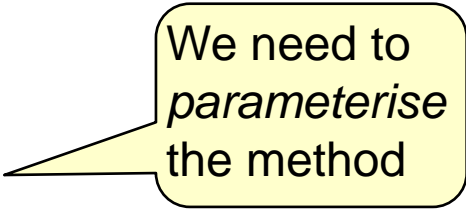
Lots of repetition.
But not exact repetition.
How can we improve it?

Good design with methods

- Key design principle:
 - Wrap up repeated sections of code into a separate method,
 - Call the method several times:

```
public void measureReactionTime ( ) {  
    this.measureQuestion( "John Quay was the Prime Minister");  
    this.measureQuestion( "6 x 4 = 23");  
    this.measureQuestion( "Summer is warmer than winter");  
    this.measureQuestion( "Wellington's population > 1,000,000 ");  
}
```

```
public void measureQuestion ( String fact ) {  
    long startTime = System.currentTimeMillis();  
    UI.askString("Is it true that " + fact . );  
    long endTime = System.currentTimeMillis();  
    UI.printf("You took %d milliseconds \n", (endTime - startTime) );  
}
```



We need to
parameterise
the method

Improving ReactionTimeMeasurer (1)

```
public void measureReactionTime() {
    this.measureQuestion("John Quay was the Prime Minister");
    this.measureQuestion("6 x 4 = 23");
    this.measureQuestion("Summer is warmer than Winter");
    this.measureQuestion("Wellington's population > 1,000,000 ");
}

public void measureQuestion(String fact) {
    long startTime = System.currentTimeMillis();
    UI.askString("Is it true that" + fact);
    long endTime = System.currentTimeMillis();
    UI.printf("You took %d milliseconds \n", (endTime - startTime) );
}
```