While folks are joining

- Get you laptops ready and login to www.crio.do
- Confirm that you are enrolled into QCalc - QCalc
- Open QCalc ME and start your workspace.
- Open Terminal and type

```
cd ~/workspace
```

- Setup Video for Reference

Crio Sprint: JAVA-112

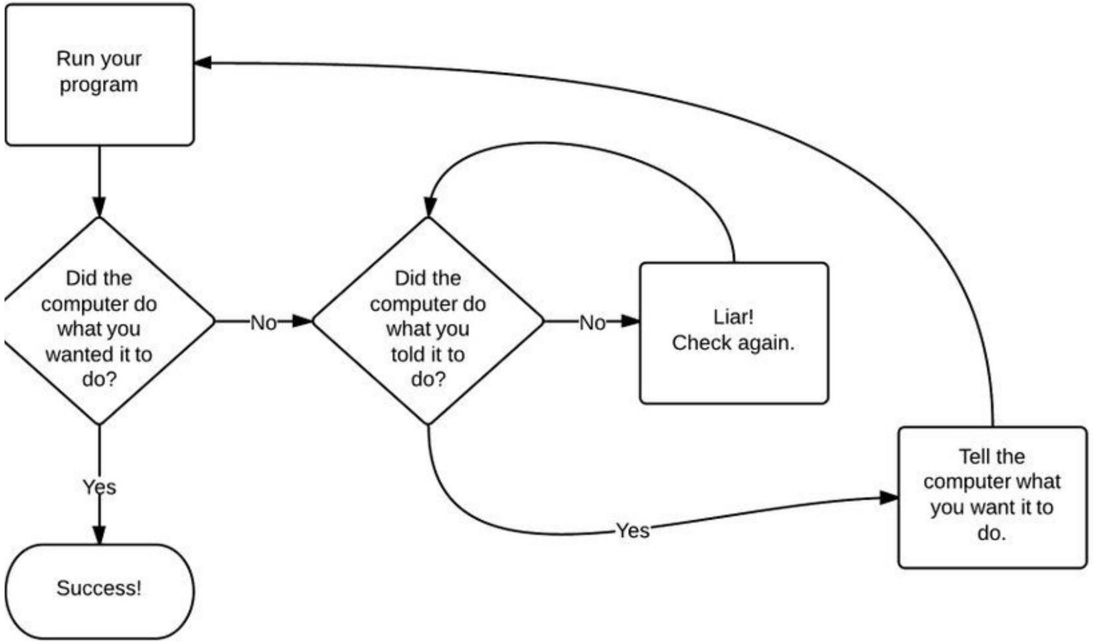
Session 9 - QCalc Inheritance

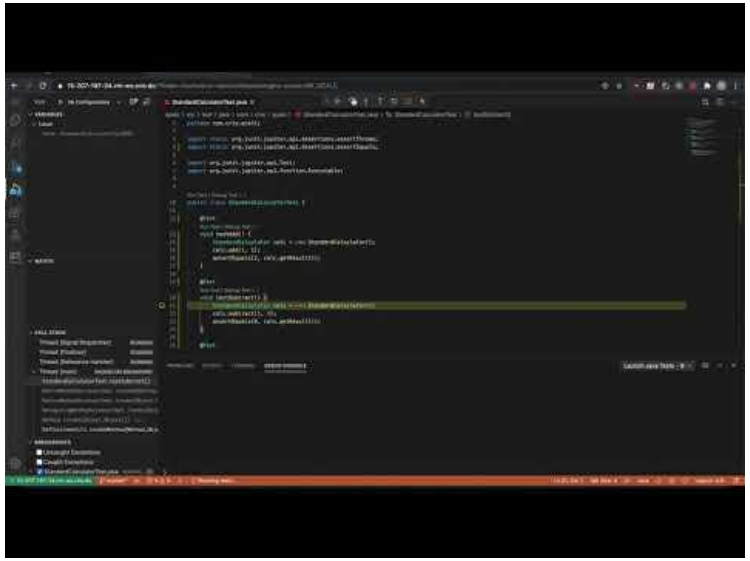
Today’s Session Agenda

- VSCode Debugger
- QCalc: Module 5 Introduction
- QCalc: Module 6 Introduction
- Buildout: XQuiz Introduction

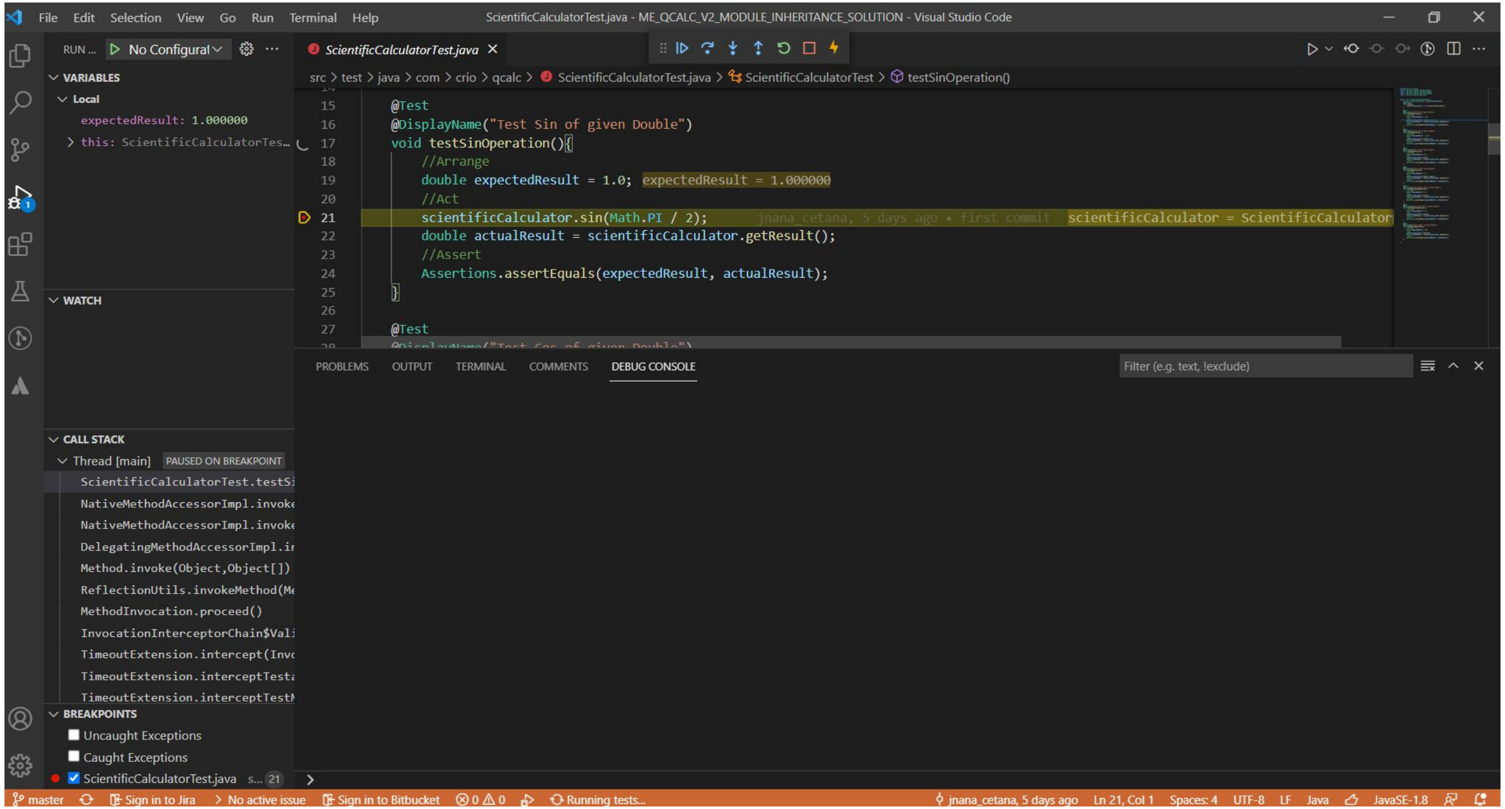
VSCode Debugger (Watch After Session)

- Learn how to debug your code with the help of Unit Tests using VSCode Debugger


Run your program
Did the computer do what you wanted it to do?
No
Did the computer do what you told it to do?
No
Liar! Check again.
Yes
Tell the computer what you want it to do.
Yes
Success!


[illegible]

VSCode Debugger Interface



ScientificCalculatorTest.java - ME_QCALC_V2_MODULE_INHERITANCE_SOLUTION - Visual Studio Code

```
expectedResult: 1.000000
> this: ScientificCalculatorTes…
```

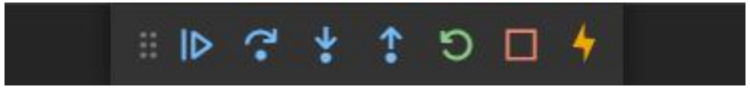
```
@Test
@DisplayName("Test Sin of given Double")
void testSinOperation(){
    //Arrange
    double expectedResult = 1.0;  expectedResult = 1.000000
    //Act
    scientificCalculator.sin(Math.PI / 2);      jnana_cetana, 5 days ago • first commit   scientificCalculator = ScientificCalculator
    double actualResult = scientificCalculator.getResult();
    //Assert
    Assertions.assertEquals(expectedResult, actualResult);
}

@Test
```

Thread [main]   PAUSED ON BREAKPOINT

```
ScientificCalculatorTest.testSi
NativeMethodAccessorImpl.invoke
NativeMethodAccessorImpl.invoke
DelegatingMethodAccessorImpl.in
Method.invoke(Object,Object[])
ReflectionUtils.invokeMethod(Me
MethodInvocation.proceed()
InvocationInterceptorChain$Vali
TimeoutExtension.intercept(Invo
TimeoutExtension.interceptTesta
TimeoutExtension.interceptTestM
```

- [ ] Uncaught Exceptions
- [ ] Caught Exceptions
- [x] ScientificCalculatorTest.java s... 21

Debugging Toolbar



Continue (F5) - This command resumes the program and continues to run it normally, without pausing unless it hits another breakpoint.

Step Over (F11) - The Step Over command takes a single step. It executes the currently highlighted line, and then pauses again.

Step Into (F10) - Step Into goes *into* that function and pauses on the first line inside.

Step Out (Shift + F11) - Step Out command executes all the code in the current function, and then pauses at the next statement (if there is one).

Stop (Shift + F5) - Stop Debugging

Restart (Ctrl + Shift + F5) - Restart Debugging

QCalc - Module 5: Extend Simple Calculator

- In this module:
  - Implement ScientificCalculator class by extending the StandardCalculator class
  - Use Math Java API library for advanced calculations
  - Override a method
  - Prevent method overriding by using final

QCalc - Module 6: Java Debugging

- In this module:
    - Fix compilation errors caused probably due to syntax / import issues.
    - Correct logical issues in the code.
    - Fix the behaviour of a method when data is invalid.
    - Write Unit Tests for edge cases initially not thought of.

Buildout: XQuiz

https://www.crio.do/learn/me/ME_BUILDOUT_XQUIZ/

What is a Buildout?

- Build-outs are independent projects that you will be building from scratch from the given higher level requirements

- Having completed an ME already, Buildouts takes you to the next level of learning experience with you planning your work and implementations

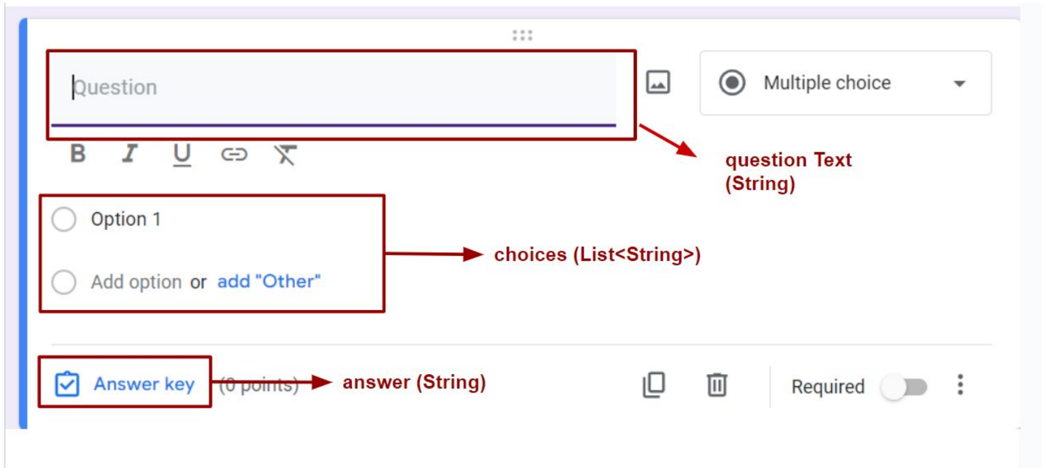
- Build-outs usually take over 15+ hours to complete. You **won't be having** further division into modules here (like in an ME)

Note:
Make sure you are well versed with the topics covered in the Bytes and tasks completed in the MEs so that you have a better experience in the Buildouts

What you will be creating

- **Question class (Question.java):** This class is a blueprint for the Question objects.
- **Quiz class (Quiz.java):** Each Quiz can have multiple Questions. This class is used as a blueprint for a Quiz object which will in turn contain multiple Question objects.
- **Application class (App.java):** This is the application class where we will be creating the Question objects and Quiz Object, displaying the quiz, evaluating the answers and printing the score along with the answer key. All of this will be done by using the attributes and methods supported in the Question.java and Quiz.java classes.

Question.java



Question

Attributes:

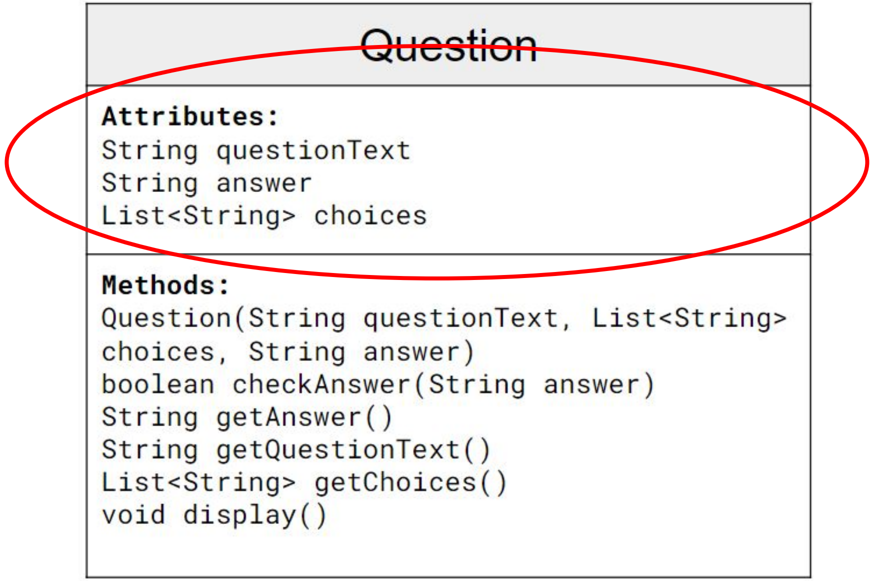
```
String questionText
String answer
List<String> choices
```

Methods:

```
Question(String questionText, List<String>
choices, String answer)
boolean checkAnswer(String answer)
String getAnswer()
String getQuestionText()
List<String> getChoices()
void display()
```

Let's build the Question class

- Defining attributes

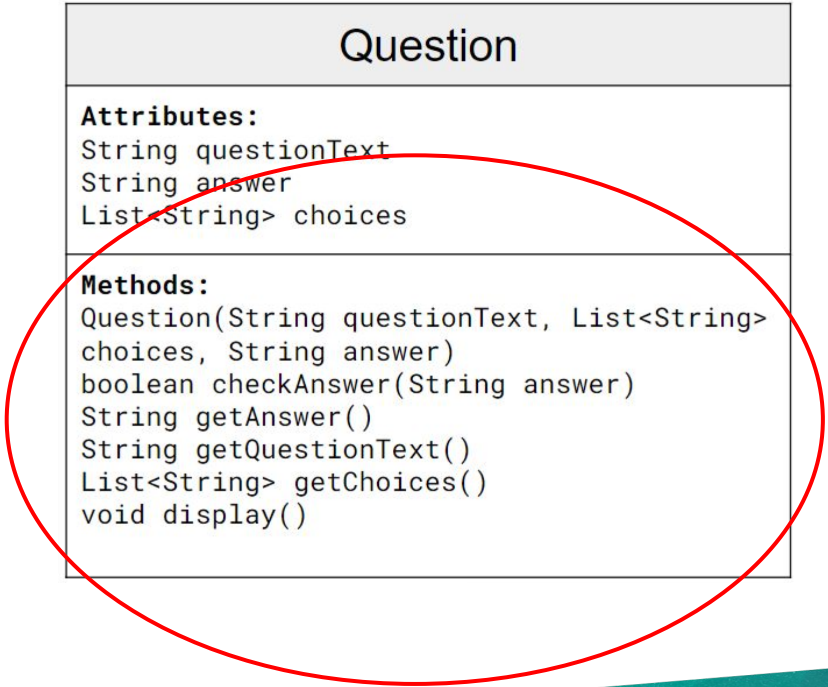

Question

Attributes:
```
String questionText
String answer
List<String> choices
```

Methods:
```
Question(String questionText, List<String>
choices, String answer)
boolean checkAnswer(String answer)
String getAnswer()
String getQuestionText()
List<String> getChoices()
void display()
```

Let's build the Question class

- Implementing methods
  - **`Question()`** constructor
  - **Getter** methods
  - **`checkAnswer()`** method



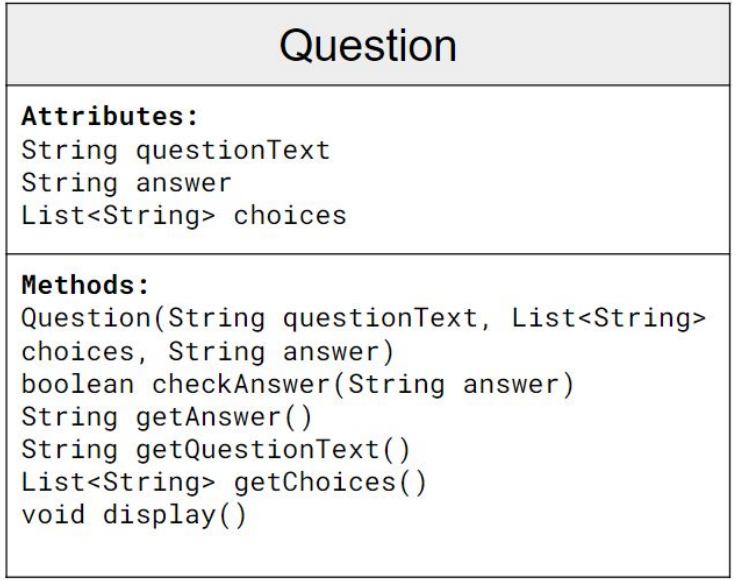
Question

Attributes:
```
String questionText
String answer
List<String> choices
```

Methods:
```
Question(String questionText, List<String>
choices, String answer)
boolean checkAnswer(String answer)
String getAnswer()
String getQuestionText()
List<String> getChoices()
void display()
```

Let's build the Question class

- Things to keep in mind
  - Use System.out.println() wherever the output needs to be displayed in the terminal.
  - The statements to be printed should be exactly the same as shown in the expected outputs, even slight changes can make your test cases fail.
  - The `display()` method is already provided in the stub, so do not change it.



Question

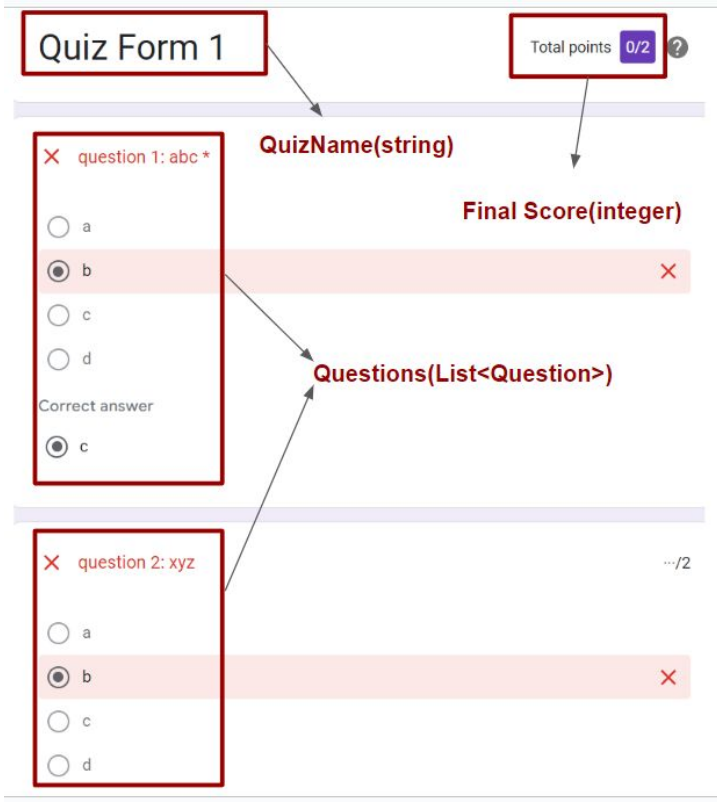
Attributes:

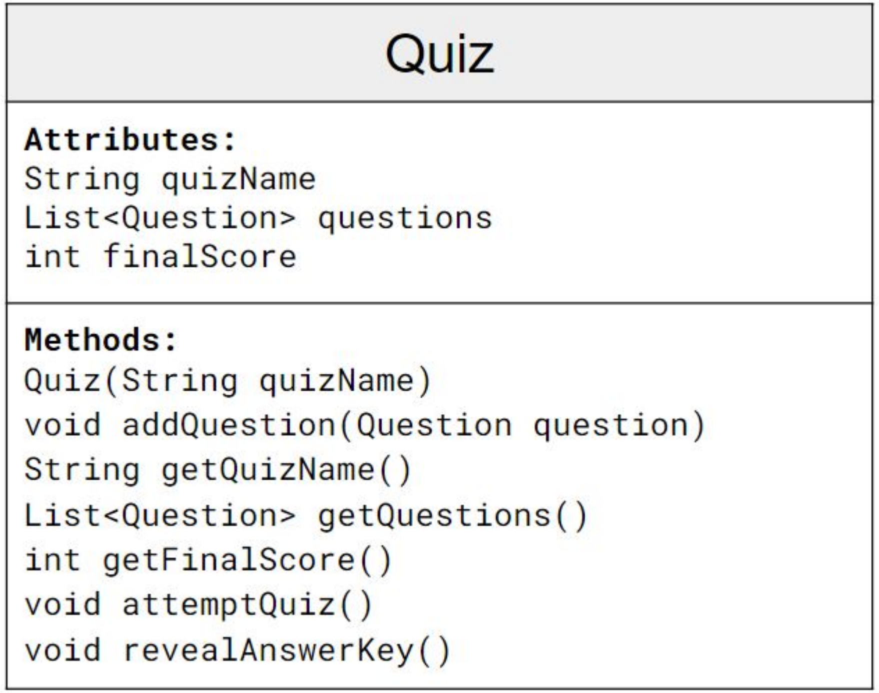
```
String questionText
String answer
List<String> choices
```

Methods:

```
Question(String questionText, List<String>
choices, String answer)
boolean checkAnswer(String answer)
String getAnswer()
String getQuestionText()
List<String> getChoices()
void display()
```

Quiz.java

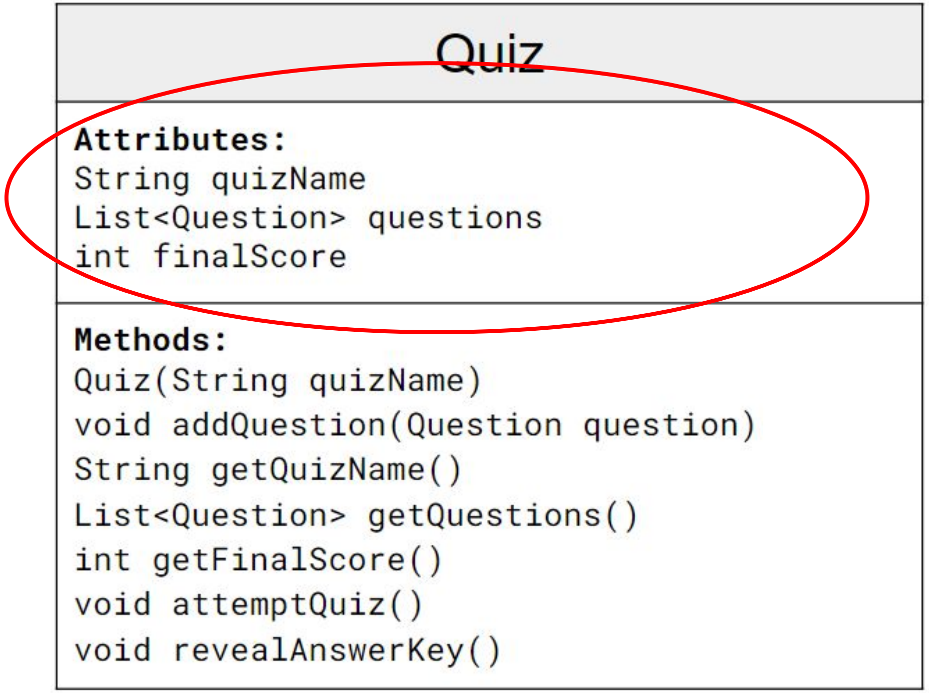




```
Quiz

Attributes:
String quizName
List<Question> questions
int finalScore

Methods:
Quiz(String quizName)
void addQuestion(Question question)
String getQuizName()
List<Question> getQuestions()
int getFinalScore()
void attemptQuiz()
void revealAnswerKey()
```

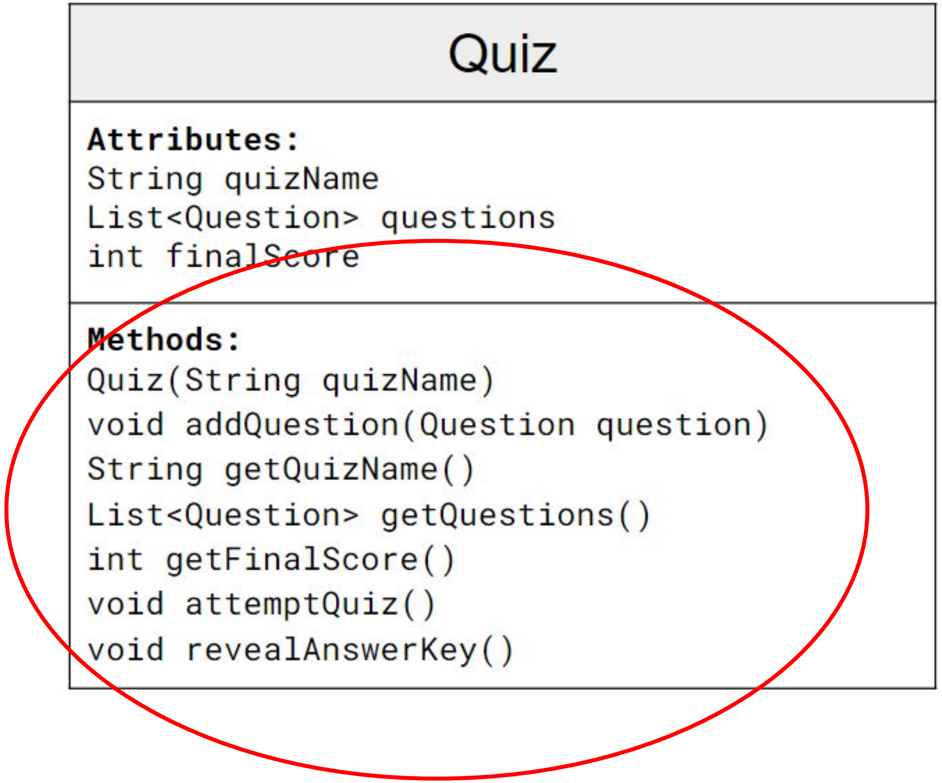
Let’s build the Quiz class

- Defining attributes



Let’s build the Quiz class

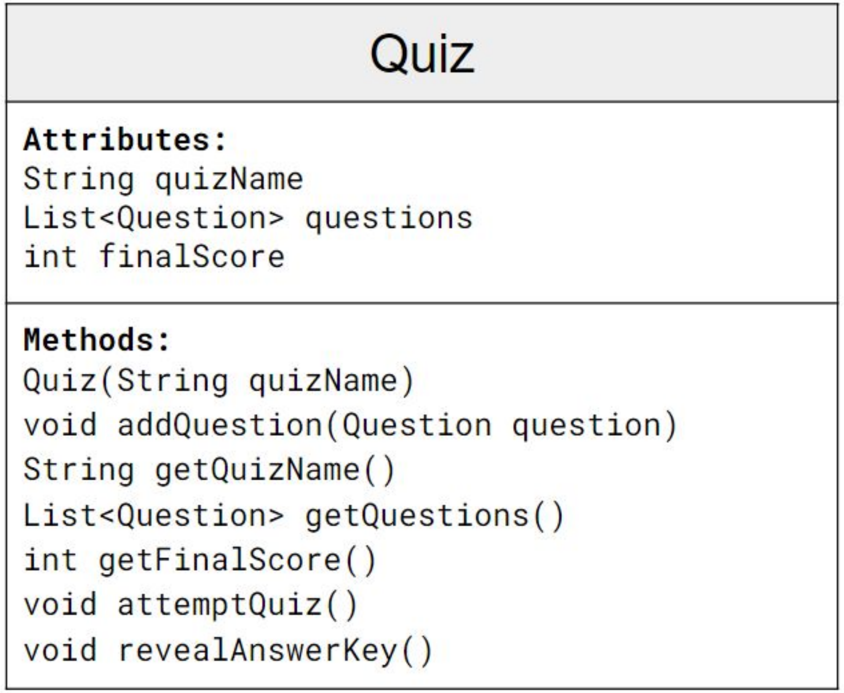
- Implementing methods
  - **Quiz()** constructor
  - **Getter** methods
  - **`addQuestion()`**
  - **`revealAnswerKey()`**



```
Quiz

Attributes:
String quizName
List<Question> questions
int finalScore

Methods:
Quiz(String quizName)
void addQuestion(Question question)
String getQuizName()
List<Question> getQuestions()
int getFinalScore()
void attemptQuiz()
void revealAnswerKey()
```

Let’s build the Quiz class

- Things to keep in mind
  - Use System.out.println() wherever the output needs to be displayed in the terminal.
  - The statements to be printed should be exactly the same as shown in the expected outputs, even slight changes can make your test cases fail.
  - The method `attemptQuiz()` is already provided in the stub, so do not change it.

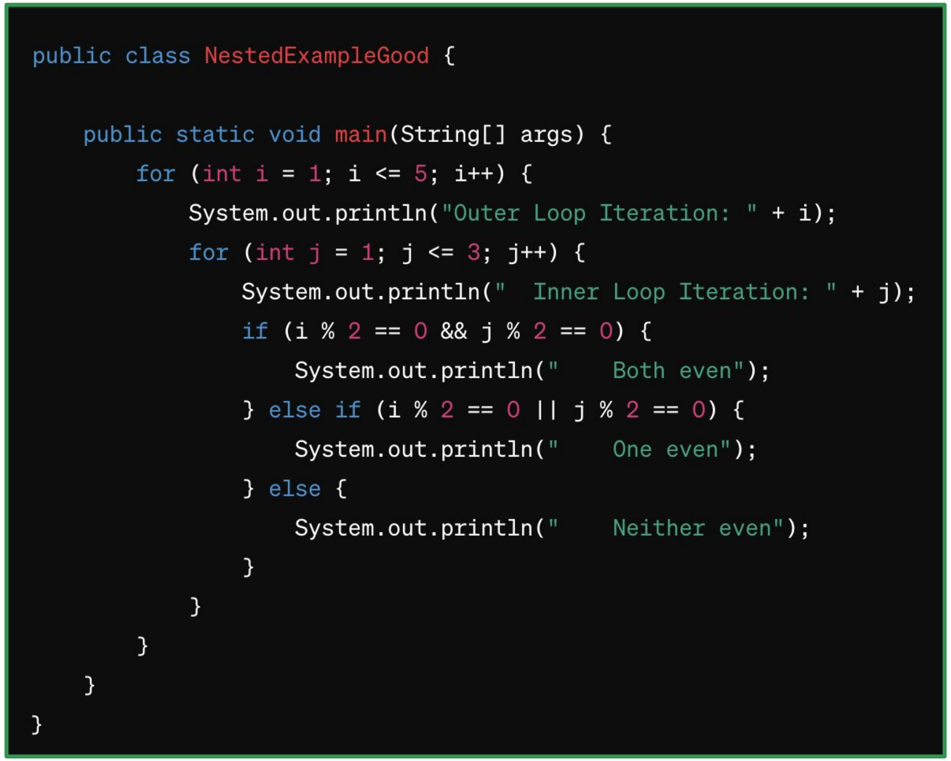


Quiz

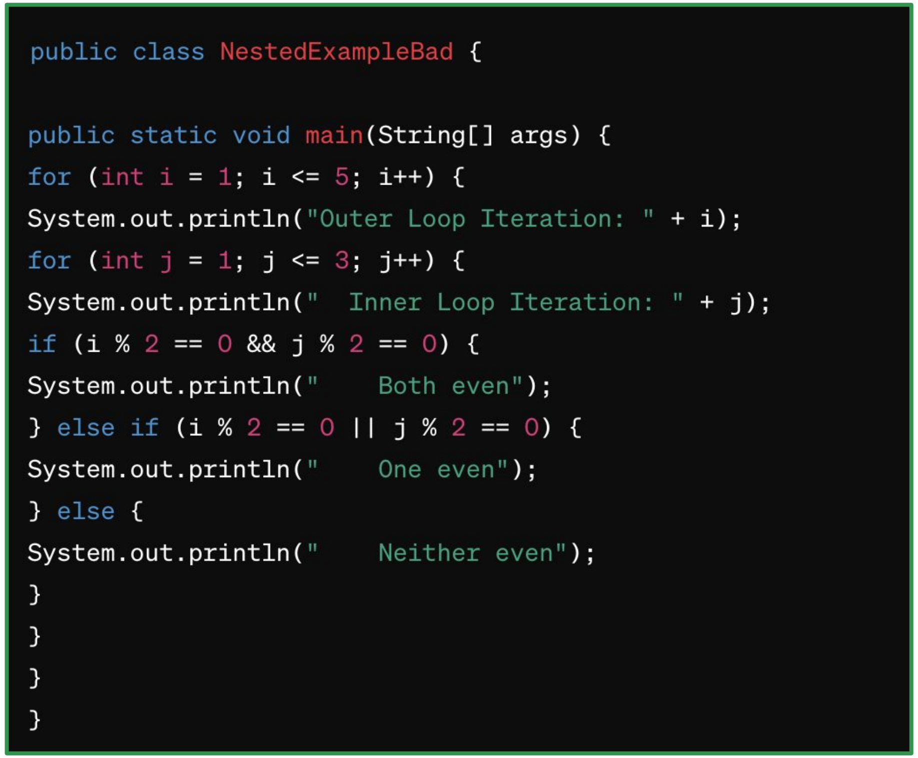
```
Attributes:
String quizName
List<Question> questions
int finalScore
```

```
Methods:
Quiz(String quizName)
void addQuestion(Question question)
String getQuizName()
List<Question> getQuestions()
int getFinalScore()
void attemptQuiz()
void revealAnswerKey()
```

Which of these programs is easily readable?



```java
public class NestedExampleGood {

    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Outer Loop Iteration: " + i);
            for (int j = 1; j <= 3; j++) {
                System.out.println("  Inner Loop Iteration: " + j);
                if (i % 2 == 0 && j % 2 == 0) {
                    System.out.println("    Both even");
                } else if (i % 2 == 0 || j % 2 == 0) {
                    System.out.println("    One even");
                } else {
                    System.out.println("    Neither even");
                }
            }
        }
    }
}
```

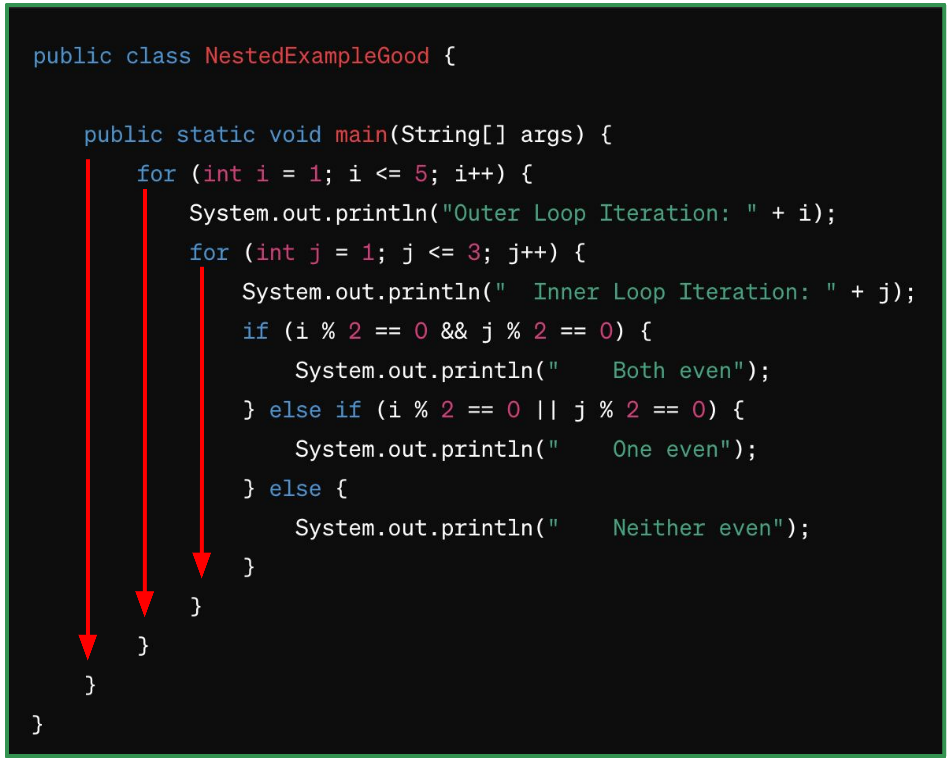


```java
public class NestedExampleBad {

public static void main(String[] args) {
for (int i = 1; i <= 5; i++) {
System.out.println("Outer Loop Iteration: " + i);
for (int j = 1; j <= 3; j++) {
System.out.println("  Inner Loop Iteration: " + j);
if (i % 2 == 0 && j % 2 == 0) {
System.out.println("    Both even");
} else if (i % 2 == 0 || j % 2 == 0) {
System.out.println("    One even");
} else {
System.out.println("    Neither even");
}
}
}
}
```

Indentation

Why we should use Indentation:

- It shows scope of each section
- Anyone (including you) reading the code can interpret the code cleanly
- It shows levels of nesting making it easier to debug



```java
public class NestedExampleGood {

    public static void main(String[] args) {
        for (int i = 1; i <= 5; i++) {
            System.out.println("Outer Loop Iteration: " + i);
            for (int j = 1; j <= 3; j++) {
                System.out.println("  Inner Loop Iteration: " + j);
                if (i % 2 == 0 && j % 2 == 0) {
                    System.out.println("    Both even");
                } else if (i % 2 == 0 || j % 2 == 0) {
                    System.out.println("    One even");
                } else {
                    System.out.println("    Neither even");
                }
            }
        }
    }
}
```

Week-3 Quiz

[Quiz Link]

Solve this quiz to access your understanding of all the session’s topics you learnt this week

Take home exercises for the session

- You will have to complete the below modules of QCalc Micro-Experience:
  - Module 5: Extend Simple Calculator
  - Module 6: Java Debugging
- You have to complete the Buildout
  - XQUIZ

Thank you