2.5 - Quiz App Alpha Release (v0.2) & Documentation

OVERVIEW

In this lesson the Strings file will be introduced and explained. Students will complete coding of any additional features for their Quiz App α-release and write a technical report. On the final day of this lesson, every student downloads and tests one another's apps and provides feedback to their peers.

Estimated Duration: 5 days

STANDARDS ADDRESSED

CSTA Standards

- 3B-AP-12: Compare and contrast fundamental data structures and their uses.
- 3B-AP-14: Construct solutions to problems using student-created components, such as procedures, modules, and/or objects.
- 3B-AP-16: Demonstrate code reuse by creating programming solutions using libraries and APIs.
- 3B-AP-17: Plan and develop programs for broad audiences using a software life cycle process.
- 3B-AP-19: Develop programs for multiple computing platforms.
- 3B-AP-22: Modify an existing program to add additional functionality and discuss intended and unintended implications.
- 3A-AP-13: Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 3A-AP-14: Use lists (arrays) to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
- 3A-AP-16: Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
- 3A-AP-21: Evaluate and refine computational artifacts to make them more usable and accessible.
- 3A-AP-23: Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

Maryland Computer Science Standards

- 12. AP.V.01 Compare and contrast foundational data structures and their primary functions.

- 12. AP.M.01 Construct solutions to problems using student-created components, such as procedures, modules, and objects to implement abstractions.
- 12. AP.M.03 Create programming solutions using libraries and APIs through the application of code reuse.
- 12. AP.PD.01 Utilize a software lifecycle process that considers security, to plan and develop programs for all types of users.
- 12. AP.PD.03 Develop different programs for various computing platforms (e.g., desktop, web, mobile).
- 12. AP.PD.06 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
- 10. AP.A.01 Develop prototypes that use algorithms (e.g., sequencing, selection, iteration, recursion, etc.) to solve computational problems by leveraging prior student knowledge and personal interest
- 10. AP.V.02 Utilize lists to simplify solutions, generalizing computational problems, instead of repeatedly utilizing simple variables.
- 10. AP.C.02 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
- 10. AP.PD.03 Evaluate and refine computational artifacts to improve usability, accessibility, and efficiency.
- 10. AP.PD.05 Represent the design elements and data flow (e.g., flowcharts, pseudocode, etc.) of the development of a complex program through the use of various visual aids and documentation techniques.

UNDERSTANDINGS

Factual Knowledge	**Procedural Knowledge**	**Conceptual Knowledge**
Students will know: • The parts of an object-oriented class • How arrays work in Java • What explicit and implicit intents are used for	*Students will be able to:* • Draw simple UML class diagrams • Write Java code for object-oriented classes • Declare, initialize, and use Java objects in client classes • Use Java data structures like arrays • Use both explicit and implicit intents correctly	*Students will understand:* • How and why we use object-oriented classes in apps • How and why we use data structures in apps

ESSENTIAL QUESTION(S)

- How can developers take advantage of object-oriented programming?
- How and when should inheritance, overloading, and overriding be used to design object-oriented software?

- How and what types of abstraction can be used to improve my code?
- What factors should we consider when determining which type of data structure to use for a specific situation?
- If I get errors while coding, how can I solve them?

OBJECTIVE

Students will code the alpha release for their Quiz App, document their work via a technical report, and test one another's apps.

ASSESSMENT / PERFORMANCE TASK

Students will submit their code and write a report on the Quiz App v0.2 (alpha), which will be graded based on the Quiz App v0.2 (Alpha) Rubric.

MATERIALS NEEDED

The basic course materials (computer, Android device, USB cable, internet access, screen projection/sharing system) are needed every day. Additionally, today you should make sure the following links all work and are accessible to students:

Readings:

- https://developer.android.com/guide/topics/resources/string-resource
- https://developer.android.com/training/basics/supporting-devices/languages
- Quiz App v0.2 (Alpha) Rubric

For the teacher:

- Updated Quiz App Sample Code on GitHub done through this lesson (Quiz App Release v0.2 alpha)
    - Does **not** include the two student-researched features

LEARNING PLAN

DAY ONE

	ACTIVITIES	SUGGESTED TIMING
2.5.1	Warm Up: Counting Strings	5 minutes
2.5.2	Strings Resource File	10 minutes
2.5.3	Work Time: Coding	40 minutes
2.5.4	Exit Ticket: Progress Check	5 minutes

DAY TWO

	ACTIVITIES	SUGGESTED TIMING
2.5.5	Warm Up: Rubric Review - Features	5 minutes
2.5.6	Work Time: Coding & Submission	50 minutes

DAY THREE

	ACTIVITIES	SUGGESTED TIMING
2.5.7	Warm Up: Rubric Review - Documentation	5 minutes
2.5.8	Work Time: Report	50 minutes

DAY FOUR

	ACTIVITIES	SUGGESTED TIMING
2.5.9	Work Time: Report & Submission	60 minutes

DAY FIVE

	ACTIVITIES	SUGGESTED TIMING
2.5.10	Peer Review Instructions	10 minutes
2.5.11	Testing & Feedback	40 minutes
2.5.12	Feedback Review & Reflection	10 minutes

ACTIVITY 2.5.1 - WARM UP (Day 1)

1. Have students count how many strings are used in their code.
    a. Include both Java String objects and any strings of text included in the XML.

ACTIVITY 2.5.2 - STRINGS RESOURCE FILE

1. Introduce the location and purpose of the strings file (under resources and under values).
    a. The *strings.xml* file keeps all text/words/sentences from one app in the same place.
    b. This makes it easier to change text that may be used in more than one place (e.g. greeting the user), as well as making it much simpler to translate the app into other languages by just swapping out a strings file (or providing both so that users may select the language) rather than having to comb through their code and find all the strings to translate/change!

2. Demonstrate finding and replacing all English strings in XML and Java with references to the strings file in a simple app (e.g. Hello, Name).
3. Point students to resources for how to include two language strings files.
4. Ask students to do this, to make their apps bilingual.
    a. You may wish to encourage students to build cross-curricular connections by using the language they are learning or have learned in high school (e.g. Spanish or French).
    b. If your students come from a variety of different countries, you may choose to encourage them to code the second strings file in another language that is spoken at home (and later, to have family members test it out).
5. **NOTE**: if you are tight on time, you should still explain the rationale and usage of the strings file, and ask students to separate out all English strings into the *strings.xml* file today, but may wish to delay adding a second language until the beta release.

ACTIVITY 2.5.3 - WORK TIME: CODING (Day 1)

1. Students work on coding their apps:
    a. adding any features not yet added from the previous lessons,
    b. adding the two strings files introduced today,
    c. making any final polishes and changes to have the app ready for its first (internal/alpha) release.
2. Example github commit with (English) strings file completed.
3. Teacher circulates and assists.
4. Daily push to GitHub!

ACTIVITY 2.5.4 - EXIT TICKET (Repeat Days 1-3)

Check in on students' progress.

ACTIVITY 2.5.5 - WARM UP (Day 2): RUBRIC - FEATURES

1. Hand out / discuss / question the students on the Quiz App v0.2 (Alpha) Rubric, focusing on the requirements in terms of features coded.
    a. Report/documentation aspects of the rubric will be discussed tomorrow.

ACTIVITY 2.5.6 - WORK TIME (Day 2): CODING & SUBMISSION

1. Students work on coding their apps, adding any remaining features and improving their apps.
2. Teacher circulates and assists.
3. Daily push to GitHub - **should be the final push of the entire finished alpha release of the app!**
    a. See note below under Day 5 concerning student builds of APK files.

ACTIVITY 2.5.7 - WARM UP (Day 3): RUBRIC - DOCUMENTATION

1. Hand out / discuss / question the students on the Quiz App v0.2 (Alpha) Rubric, focusing on the documentation requirements.

ACTIVITY 2.5.8 - WORK TIME (Day 3): REPORT

1. Students work on writing their technical reports.
2. Teacher circulates and assists.

ACTIVITY 2.5.9 - WORK TIME (Day 4): REPORT & SUBMISSION

1. Students work on writing their technical reports (**due today**).
2. Teacher circulates and assists.
3. Students turn in their reports.

ACTIVITY 2.5.10-12 - PEER REVIEW, TESTING, FEEDBACK, & REFLECTION (Day 5)

1. Links to classmates' apps:
    a. Compile links to each GitHub repository in a shared document (e.g. Google Docs); you will provide this document to all students and they can clone the GitHub repo and download the apps via Android Studio to their devices to test them out.
    b. Another option, if you don't want students' to have access to each other's code, or if you want to practice finalizing an app for release, would be to have students build the APK file and submit that on Day 2, and just provide links to the APK files in the shared doc.
    c. A third option, not recommended, is for you to do the downloading work on each student device prior to today's class.
2. Several factors here depend on your class size. For example, with a small class, you may wish to have each student review and provide feedback on every other student's app. While in a larger class, you may want to randomly shuffle and assign each student 3-5 apps to review.
3. One possible benefit of testing is to test out the apps on multiple different devices and see how it acts differently depending on the make/model of device and version of Android.
    a. If students have their own personal Android devices (vs. a class set), you may wish to encourage them to use those devices in testing today.
    b. Or if you have a few additional devices that are for example, donations of old Android phones and tablets, you may hand those out to students for testing beyond the standardized class set they usually use.
4. It is recommended to create a feedback Google Form (or comparable, e.g. SurveyMonkey) that students will answer several standard questions reviewing the quality of their peers' work on the different Quiz Apps, while also providing open-ended questions for advice and qualitative feedback.
    a. Here is a sample Google Doc you may use/adapt: Sample Quiz App Alpha Test Feedback Form

    b. With a Google Form, for example, you can then sort the attached spreadsheet and copy/paste the feedback directed at each student into their own spreadsheet which they can use for reflections.
    c. An alternative could be a paper questionnaire, with enough copies for each student/app combo (i.e. (n)*(n-1) copies for a class of n students). Then they turn them in as they complete them and you can return to the student being reviewed.
5. For the day's class flow:
    a. Review your instructions for the students,
    b. Provide time to download and test and provide feedback to one another,
    c. Then ask them each to read the feedback from their peers about their app.
    d. As they read their feedback, ask them to identify 2-3 concrete action steps to make their apps better.

HOMEWORK

Assign students to watch videos on Udacity about OOP for Android (User Input course, Units 4 & 5).
NOTE: This needs to be complete before Lesson 1.6.