***Course Submission Cover Sheet Module: CC4001 Programming Engineering***

***Component no: 003***

***Weighting: 60% of module mark***

***Deadline: 1st of May 2024***

**Module Leader: Sandra Fernando Student ID:**

PLAGIARISM

You are reminded that there exist regulations concerning plagiarism. Extracts from these regulations are printed below. Please sign below to say that you have read and understand these extracts:

(Signature:) ddt Date:

This header sheet should be attached to the work you submit. No work will be accepted without it.

Extracts from University *Regulations on*

Cheating, Plagiarism and Collusion

Section 2.3: "The following broad types of offence can be identified and are provided as indicative examples ….

1. **Cheating: including taking unauthorised material into an examination; consulting unauthorised material outside the examination hall during the examination; obtaining an unseen examination paper in advance of the examination; copying from another examinee; using an unauthorised calculator during the examination or storing unauthorised material in the memory of a programmable calculator which is taken into the examination; copying coursework.**
2. **Falsifying data in experimental results.**
3. Personation, where a substitute takes an examination or test on behalf of the candidate. Both candidate and substitute may be guilty of an offence under these Regulations.
4. **Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.**
5. Collusion to present joint work as the work solely of one individual.
6. Plagiarism, where the work or ideas of another are presented as the candidate's own.
7. Other conduct calculated to secure an advantage on assessment.
8. Assisting in any of the above.

Some notes on what this means for students:

1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.
2. Taking extracts from published sources *without attribution* is an offence. To quote ideas, sometimes using extracts, is generally to be encouraged. Quoting ideas is achieved by stating an author's argument and attributing it, perhaps by quoting, immediately in the text, his or her name and year of publication, e.g. “e = mc2 (Einstein 1905)". A *references* section at the end of your work should then list all such references in alphabetical order of authors' surnames. (There are variations on this referencing system which your tutors may prefer you to use.) If you wish to quote a paragraph or so from published work then indent the quotation on both left and right margins, using an italic font where practicable, and introduce the quotation with an attribution.

1

Contents

[**1. GitHub Repository: 2**](#_Toc165394751)

[**2. Class Diagram: 2**](#_Toc165394752)

[**3.Method Descriptions: 3**](#_Toc165394753)

[**4. Pseudocode for Button-handling Methods: 6**](#_Toc165394754)

[**5. Textboxes, Input Check with try/catch: 9**](#_Toc165394755)

[**6. Buttons and Action Performed Methods: 10**](#_Toc165394756)

[**7. Testing in Command Prompt: 10**](#_Toc165394757)

[**8. GUI: 11**](#_Toc165394758)

[**9. Testing Evidence: 11**](#_Toc165394759)

[**11. New Features 15**](#_Toc165394760)

[**12. Conclusion: 16**](#_Toc165394761)

**Gadget Shop Project Report**

# 1. GitHub Repository:

- GitHub Repository: [GadgetShop Project Repository] (https://github.com/your/repository)

# **2. Class Diagram:**

A screenshot of a computer

Description automatically generated

# 3.Method Descriptions:

**1. Gadget.java:**

* **Gadget(String model, double price, int weight, String size)**:
  + *Description:* This constructor initializes a Gadget object with specific attributes like model, price, weight, and size.
  + *Details:* It takes parameters for the model (a string), price (a floating-point number), weight (an integer), and size (a string) and assigns them to the respective fields of the Gadget object.
* **getModel(): String**:
  + *Description:* This method retrieves the model of the gadget.
  + *Details:* It returns a string representing the model of the gadget.
* **setModel(String model): void**:
  + *Description:* This method sets the model of the gadget to a specified value.
  + *Details:* It takes a string parameter representing the new model and updates the model of the gadget accordingly.
* **getPrice(): double**:
  + *Description:* This method retrieves the price of the gadget.
  + *Details:* It returns a floating-point number representing the price of the gadget.
* **setPrice(double price): void**:
  + *Description:* This method sets the price of the gadget to a specified value.
  + *Details:* It takes a floating-point number parameter representing the new price and updates the price of the gadget accordingly.
* **getWeight(): int**:
  + *Description:* This method retrieves the weight of the gadget.
  + *Details:* It returns an integer representing the weight of the gadget.
* **setWeight(int weight): void**:
  + *Description:* This method sets the weight of the gadget to a specified value.
  + *Details:* It takes an integer parameter representing the new weight and updates the weight of the gadget accordingly.
* **getSize(): String**:
  + *Description:* This method retrieves the size of the gadget.
  + *Details:* It returns a string representing the size of the gadget.
* **setSize(String size): void**:
  + *Description:* This method sets the size of the gadget to a specified value.
  + *Details:* It takes a string parameter representing the new size and updates the size of the gadget accordingly.
* **display(): void**:
  + *Description:* This method displays the details of the gadget.
  + *Details:* It prints out the model, price, weight, and size of the gadget to the console.

2. Mobile.java:

* **Mobile(String model, double price, int weight, String size, int credit)**:
  + *Description:* This constructor initializes a Mobile object with specific attributes like model, price, weight, size, and credit.
  + *Details:* It takes parameters for the model, price, weight, size, and credit, and assigns them to the respective fields of the Mobile object.
* **getCredit(): int**:
  + *Description:* This method retrieves the credit balance of the mobile.
  + *Details:* It returns an integer representing the credit balance of the mobile.
* **setCredit(int credit): void**:
  + *Description:* This method sets the credit balance of the mobile to a specified value.
  + *Details:* It takes an integer parameter representing the new credit balance and updates the credit balance of the mobile accordingly.
* **makeCall(String phoneNumber, int duration): void**:
  + *Description:* This method initiates a call from the mobile to a specified phone number for a specified duration.
  + *Details:* It takes parameters for the phone number and call duration, and simulates making a call from the mobile.

3. MP3.java:

* **MP3(String model, double price, int weight, String size, double memory)**:
  + *Description:* This constructor initializes an MP3 object with specific attributes like model, price, weight, size, and memory.
  + *Details:* It takes parameters for the model, price, weight, size, and memory, and assigns them to the respective fields of the MP3 object.
* **getMemory(): double**:
  + *Description:* This method retrieves the memory capacity of the MP3 player.
  + *Details:* It returns a floating-point number representing the memory capacity of the MP3 player.
* **setMemory(double memory): void**:
  + *Description:* This method sets the memory capacity of the MP3 player to a specified value.
  + *Details:* It takes a floating-point number parameter representing the new memory capacity and updates the memory capacity of the MP3 player accordingly.
* **downloadMusic(double downloadSize): void**:
  + *Description:* This method downloads music to the MP3 player with a specified download size.
  + *Details:* It takes a parameter for the download size and simulates downloading music to the MP3 player with the specified size.

# 4. Pseudocode for Button-handling Methods:

**Getting the display number from the GUI:**

Convert GUI displayNumber Field value to Integer

This pseudocode retrieves the display number entered in the GUI, aligning with the concept of GUI interaction.

**Adding a mobile:**

try

Get model, price, weight, size, and credit from GUI text fields.

Create a new Mobile object with the obtained values.

Add the Mobile object to the gadgets ArrayList.

Add the model to the deviceNames ArrayList.

Add the model to the deviceComboBox.

Display success message.

catch (NumberFormatException)

Display error message for invalid input.

This pseudocode corresponds to the addMobile() method in the GadgetShop class, which gathers data from the GUI fields and creates a new Mobile object.

**Adding an MP3:**

try

Get model, price, weight, size, and memory from GUI text fields.

Create a new MP3 object with the obtained values.

Add the MP3 object to the gadgets ArrayList.

Add the model to the deviceNames ArrayList.

Add the model to the deviceComboBox.

Display success message.

catch (NumberFormatException)

Display error message for invalid input.

This aligns with the addMP3() method in the GadgetShop class, which collects data from the GUI fields and creates a new MP3 object.

**Displaying all gadgets in the array list:**

if gadgets ArrayList is empty

Display message: "No gadgets to display."

else

Iterate over gadgets ArrayList.

Display details of each gadget

This pseudocode corresponds to the displayAll() method in the GadgetShop class, which shows details of all gadgets stored in the ArrayList.

**Making a call:**

if gadgets ArrayList is empty

Display message: "No gadgets available to make a call."

else

Get selected device index from deviceComboBox

if selected device index is -1

Display error message: "Please select a device first!"

else

Get mobile object from gadgets ArrayList using selected device index.

Get phone number and duration from GUI text fields.

Try

Convert duration to Integer.

Make call using mobile object with phone number and duration.

Display success message.

catch (NumberFormatException)

Display error message for invalid input for duration.

catch (ClassCastException)

Display error message: "Selected device is not a mobile!"

This pseudocode corresponds to the makeACall() method in the GadgetShop class, which handles making calls from mobile devices.

**Downloading music:**

if gadgets ArrayList is empty

Display message: "No gadgets available to download music."

else

Get selected device index from deviceComboBox.

if selected device index is -1

Display error message: "Please select a device first!"

else

Get MP3 object from gadgets ArrayList using selected device index.

Get download size from GUI text field.

Try

Convert download size to Double.

Download music to MP3 object with download size.

Display success message.

catch (NumberFormatException)

Display error message for invalid input for download size.

catch (ClassCastException)

Display error message: "Selected device is not an MP3 player!"

This pseudocode aligns with the downloadMusic() method in the GadgetShop class, which handles downloading music to MP3 devices.

# 5. Textboxes, Input Check with try/catch:

* Textboxes are used in the GUI for inputting values such as model, price, weight, size, credit, memory, phone number, duration, and download size.
* Input validation is performed using try/catch blocks to ensure that only valid numeric values are accepted for fields where numeric input is expected. If an invalid input is detected, an error message is displayed to the user.

# 6. Buttons and Action Performed Methods:

* Buttons in the GUI include "Add Mobile", "Add MP3", "Clear", "Display All", "Make a Call", and "Download Music".
* Corresponding ActionPerformed methods are implemented in the GadgetShop class to handle the actions performed when these buttons are clicked.

# 7. Testing in Command Prompt:

* The program can be compiled and run using the command prompt by navigating to the directory containing the compiled **.class** files and running the command **java GadgetShop**.
* Screenshot:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# 8. GUI:

* The GUI includes components for displaying gadgets, adding mobile phones and MP3 players, making calls, and downloading music.
* Screenshot: A screenshot of a computer

  Description automatically generated

# 9. Testing Evidence:

* Test 1: Screenshot of Adding a Mobile

A screenshot of a computer

Description automatically generated

* Test 2: Screenshot of Adding an MP3 Player

A screenshot of a computer

Description automatically generated

* Test 3: Screenshot of Displaying All Gadgets

A screenshot of a computer

Description automatically generated

* Test 4: Screenshot of Making a Call

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Test 5: Screenshot of Downloading Music

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Test 6: Screenshot of Running in Command Prompt

A screenshot of a computer

Description automatically generated

* Test 7: Screenshot of Dialog Box for Unsuitable Values

A screenshot of a computer

Description automatically generated

**10) Error Detection and Correction**

1. Error: Invalid input values causing NumberFormatException. Detection: Handled using try/catch blocks. Correction: Display error message and prompt for valid input.
2. Error: Selecting incorrect gadget type for actions like making a call or downloading music. Detection: Checked gadget type before performing actions. Correction: Display appropriate error message.
3. Error: Attempting to perform actions on an empty gadgets list. Detection: Checked if gadgets list is empty before performing actions. Correction: Display appropriate error message.

# 11) New Features

**Additional Features Implemented in the Report:**

**1. Text Field Input Methods**

* Implemented methods to read the contents of each text field and return the corresponding value.
* Methods handle conversion to the correct numerical data type for numeric fields.
* Added specific input methods for model, size, phone number, duration, download size, and display number.
* Implemented input validation using try/catch statements for the display number input field.

**2. Buttons**

* Added functionality for buttons to handle different actions:
  + **Add Mobile**: Create a new Mobile object with input values and add it to the array list of gadgets.
  + **Add MP3**: Create a new MP3 object with input values and add it to the array list of gadgets.
  + **Clear**: Clear text from all text fields.
  + **Display All**: Display details of all gadgets in the array list.
  + **Make A Call**: Perform a call with input phone number and duration if a valid display number is selected.
  + **Download Music**: Download music with input download size if a valid display number is selected.

**3. Error Handling**

* Implemented error handling for input validation and user interaction.
* Display appropriate error messages for invalid input values or out-of-range display numbers.
* Utilized try/catch statements to handle exceptions gracefully.

**Conclusion**

The additional features enhance the functionality and usability of the GadgetShop application. Input methods ensure proper data handling, buttons enable intuitive interaction, and error handling provides a smooth user experience. Through these enhancements, the application becomes more robust and user-friendly, offering a comprehensive solution for managing gadgets effectively.

# 12. Conclusion:

* In conclusion, the GadgetShop project has provided valuable insights into Java programming, GUI development, and error handling.
* This assignment enhanced understanding of object-oriented principles, including inheritance and polymorphism, and improved skills in implementing graphical user interfaces.
* Overall, the project served as a practical application of concepts learned in the course, contributing to a deeper understanding of software development processes.