



Year
2023-24

CS4001 Programming coursework



London Metropolitan University

Student ID: 21049834

Module Tutor: Sahar Al-Sudani & Sandra
Fernando.

Module: CS4001 Programming

Section 1.	UML Class Diagram.....	3
Section 1.1.	GitHub Link.....	3
Section 2.	GUI.....	4
Section 3.	Functionality and Pseudocode.....	5
3.1.	GadgetshopGUI class (<i>public void actionPerformed(ActionEvent event)</i>) method.....	5
3.2.	GadgetshopGUI class (<i>public void addMobile()</i>) method and Pseudocode.....	6
3.3.	GadgetshopGUI class (<i>public void addM3()</i>) method and Pseudocode.....	6
3.4.	GadgetshopGUI class (<i>public void makeCall()</i>) method and Pseudocode.....	7
3.5.	GadgetshopGUI class (<i>public void viewAll()</i>) method and Pseudocode.....	7
3.6.	GadgetshopGUI class (<i>public void clear()</i>) method and Pseudocode.....	8
3.7.	GadgetshopGUI class (<i>public int getCredit()</i>) method and Pseudocode.....	9
3.8.	GadgetshopGUI class (<i>private Double getprice()</i>) method and Pseudocode.....	9
3.9.	GadgetshopGUI class (<i>private int getweight()</i>) method and Pseudocode.....	10
4.	GadgetshopGUI class (<i>private String getsize()</i>) method and Pseudocode.....	10
4.1.	GadgetshopGUI class (<i>public int getaModel()</i>) method and Pseudocode.....	11
4.2.	GadgetshopGUI class (<i>public int getAvailableMemory()</i>) method and Pseudocod.....	11
4.3.	GadgetshopGUI class (<i>public int get getDisplayNumber()</i>) method and Pseudocode.....	12
4.4.	GadgetshopGUI class (<i>public int getDurationMinutes()</i>) method and Pseudocode.....	13
4.5.	GadgetshopGUI class (<i>public String getPhoneNumber()</i>) method and Pseudocode.....	13
4.6.	Gadget class method and Pseudocode.....	14
4.7.	GadgetshopGUI class Gadget class, (<i>public Gadget (String model, double price, int weight, String size)</i>) method and Pseudocode.....	14
4.8.	Gadget class, (<i>public String getModel()</i>) method and Pseudocode.....	15
4.9.	Gadget class, (<i>public double getPrice()</i>) method and Pseudocode.....	15
5.	Gadget class, (<i>public int getWeight()</i>) method and Pseudocode.....	16
5.1.	Gadget class, (<i>public String getSize()</i>) method and Pseudocode.....	16
5.2.	Gadget class, (<i>public String getSize()</i>) method and Pseudocode.....	17
5.3.	Gadget class, (<i>public void display()</i>) method and Pseudocode.....	17
5.4.	Mobile class, (<i>public Mobile(String model, double price, int weight, String size, int callingCredit)</i>) method and Pseudocode.....	18
5.5.	Mobile class, (<i>public int getCallingCredit()</i>) method and Pseudocode.....	18

5.6.	Mobile class, (<i>public void addCallingCredit(int creditToAdd)</i>) method and Pseudocode.....	19
5.7.	Mobile class, (<i>public void makeCall(String phoneNumber, int durationMinutes)</i>) method and Pseudocode.....	19
5.8.	Mobile class, (<i>public void display()</i>) method and Pseudocode.....	20
5.9.	MP3 class, (<i>public MP3(String model, double price, int weight, String size, double availableMemory)</i>) method and Pseudocode.....	20
6.	MP3 class, (<i>public double getAvailableMemory()</i>) method and Pseudocode.....	21
6.1.	MP3 class, (<i>public void downloadMusic(double memoryUsed)</i>) method and Pseudocode...	21
6.2.	MP3 class, (<i>public void display()</i>) method and Pseudocode.....	22
Section 7.	Testing the functionality of the program.....	22
Test 1:	Adding a mobile to the array list.....	22
Test 2:	Adding an MP3 Player to the array list.....	22
Test 3:	Displaying all gadgets stored in the array list.....	23
Test 4:	Making a call.....	23
Test 5:	Compiling and running the program from the command prompt.....	23
Test 6:	Appropriate dialogue appears when an unsuitable value is entered.....	24
Test 7:	An appropriate message appears when attempting to make a call with insufficient credit.....	24
Section 8.	Error detection and error correction.....	24
Case 8.1.	Syntax error and correction.....	24
Case 8.2.	GUI error and correction.....	24
Case 8.3.	Download Music error.....	25
Conclusion.....		25

Section 1. UML Class Diagram.

This section will guide you through the logical flow of our software's structure, helping you understand how different classes interact with each other.

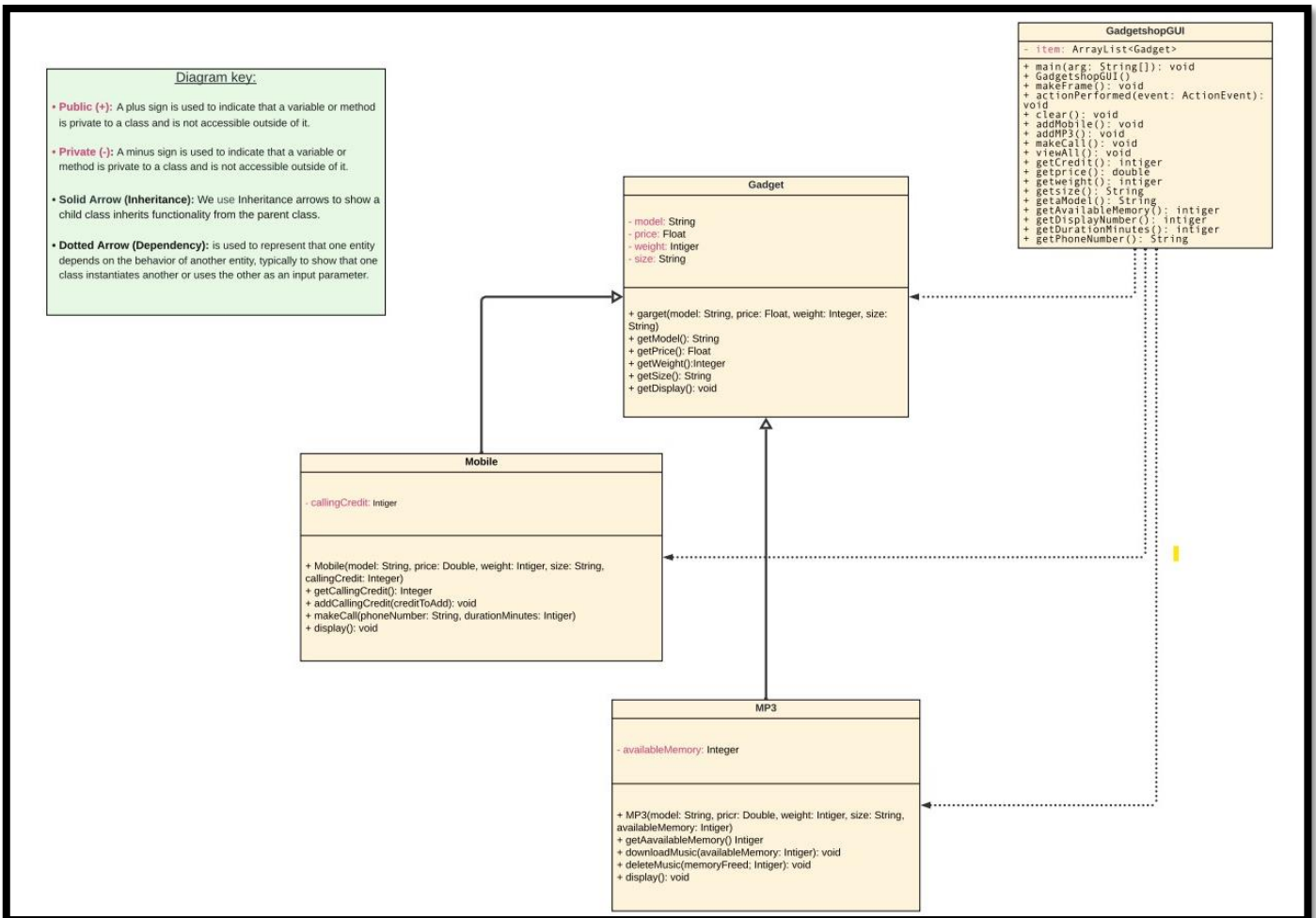


Diagram key:

Public (+): A plus sign is used to indicate that a variable or method is public to a class and is accessible outside of it.

Private (-): A minus sign indicates that a variable or method is private to a class and is not accessible outside of it.

Solid Arrow (Inheritance): We use Inheritance arrows to show a child class inherits functionality from the parent class.

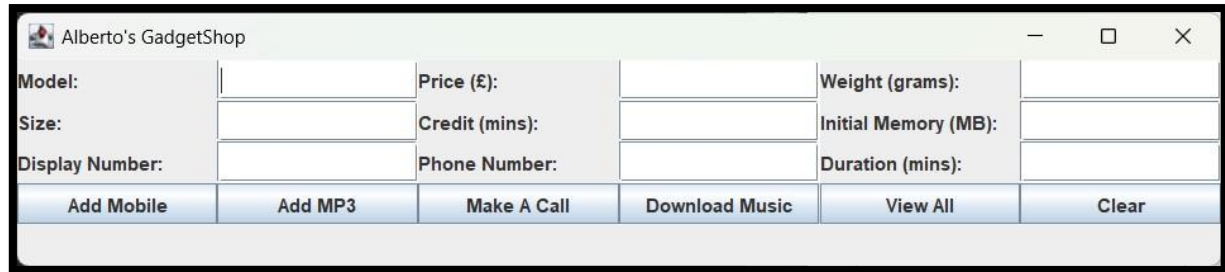
Dotted Arrow (Dependency): This symbol represents the dependence of one entity on the behaviour of another, typically to show that one class instantiates another or uses the other as an input parameter.

Section 1.1. GitHub Link.

<https://github.com/AlbertoNovas05/Gadgetshop-coursework>

Section 2. GUI.

During class, my teacher Sandra Fernando suggested a technique for designing a graphical user interface (GUI): to begin by sketching it on paper. Following her advice, I created a simple design similar to what we had done during a workshop session. This is what I came up with.



I used a GridLayout with a 5x8 configuration to create the desired layout for my contentPane. To achieve the final design, I added buttons, labels, and a text box. The JFrame is resizable, allowing the user to adjust the window size based on their preference.

The GridLayout function generates a layout that displays each component in a separate column within a row.

The panel contain 9 JLabels, 9 JField, and 6 buttons.

Labels:

- modelLabel
- priceLabel
- weightLabel
- sizeLabel
- creditLabel
- availableMemoryLabel
- displayNumberLabel
- durationLabel

Fields:

- modelField
- priceField
- weightField
- sizeField
- creditField
- availableMemoryField
- displayNumberField
- durationField

Buttons:

- addMobileButton
- addMP3button
- makeCallButton
- downloadMusic
- viewAllButton
- clearButton

Section 3. Functionalities and Pseudocodes.

In Java programming language, the term "functionality" usually refers to the behaviour of methods or functions. In Java, a function is a block of code that is designed to carry out a specific task. It helps to organise code, making it more modular and reusable. Functions in Java are always a part of a class. In order to use them, you need to call the function using the class or an object of the class.

Pseudocode is a straightforward and high-level depiction of an algorithm or code. It is written in plain English and doesn't follow any specific syntax rules, making it easy for anyone to comprehend, irrespective of their programming background. The primary use of pseudocode is to plan and visualise the logic of a program before writing the actual code.

3.1. GadgetshopGUI class (public void actionPerformed(ActionEvent event)) method.

Once the user interface design has been completed, it is necessary to implement its functionality. To do so, we need to utilise an ActionListener.

This ActionListener helps manage action events triggered by user interactions with graphical user interface (GUI) components. In my case, I am adding actions to the buttons.

```
{
    // Get the action command from the event
    String command = event.getActionCommand();
    if(command.equals("Add Mobile")) // Perform actions based on the command
    {
        addMobile();// Call the addMobile method
    }

    if(command.equals("Add MP3"))
    {
        addMP3();// Call the addMP3 method
    }

    if(command.equals("Make A Call"))
    {
        makeCall();// Call the makeCall method
    }

    if(command.equals("View All"))
    {
        viewAll();// Call the viewAll method
    }

    if(command.equals("Clear"))
    {
        clear();// Call the clear method
    }
}
```

Pseudocode:

Define a method:

Get the action command from the event

If the command is "Add Mobile":

Perform the addMobile method

If the command is "Add MP3":

Perform the addMP3 method

If the command is "Make A Call":

Perform the makeCall method

If the command is "View All":

Perform the viewAll method

If the command is "Clear":

Perform the clear method

End of method

3.2. GadgetshopGUI class (public void addMobile()) method and Pseudocode.

This method addMobile() is responsible for adding a new mobile gadget to a list of items. Here's a breakdown of what it does:

The following code creates a new Mobile object using the given attributes (model, price, weight, size, and credit) and adds it to a list of items. A success message is then displayed to notify the user that the new mobile gadget was added successfully.

```
/**
 * Method to add a new mobile gadget to the list of items.
 */
public void addMobile()
{
    // Create a new Mobile object with the provided attributes
    Mobile newMobile = new Mobile(getModel(), getprice(), getweight(), getsize(), getCredit());
    item.add(newMobile); // Add the new mobile gadget to the list of items
    // Display a success message
    JOptionPane.showMessageDialog(frame, "A new mobile has been successfully added.");
}
```

Pseudocode:

Define a method called addMobile:

Create a new object 'newMobile' of class 'Mobile' with attributes from the following

methods: getModel(), getprice(), getweight(), getsize(), getCredit()

Add 'newMobile' to the list 'item'

Display a message box with the message "A new mobile has been successfully added."

End of method

3.3. GadgetshopGUI class (public void addMP3()) method and Pseudocode.

This addMP3() method creates a new MP3 object with the provided attributes, adds it to a list of items, and then displays a success message to the user. Then it tells the user that the MP3 player was successfully added.

```
/**
 * Method to add a new MP3 gadget to the list of items.
 */
public void addMP3()
{
    // Create a new MP3 object with the provided attributes
    MP3 newMP3 = new MP3(getModel(), getprice(), getweight(), getsize(), getAvailableMemory());
    item.add(newMP3); // Add the new MP3 gadget to the list of items
    // Display a success message
    JOptionPane.showMessageDialog(frame, "A new MP3 has been successfully added.");
}
```

Pseudocode:

Define a method called addMP3:

Create a new object 'newMP3' of class 'MP3' with attributes from the following methods:

getModel(), getprice(), getweight(), getsize(), getAvailableMemory()

Add 'newMP3' to the list 'item'

Display a message box with the message "A new MP3 has been successfully added."

End of method

3.4. GadgetshopGUI class (public void makeCall()) method and Pseudocode.

This method aims to initiate a call using a mobile device. If the device is not a mobile or the displayed number is incorrect, it will inform the user that an error has occurred.

In simple terms, this method attempts to make a call using a mobile gadget. If the gadget is not a mobile or the display number is invalid, it tells the user that an error occurred.

```
public void makeCall()
{
    Gadget gadget = item.get(getDisplayNumber()); // Retrieve the gadget based on the display number
    // Check if the display number is valid and the gadget is a Mobile
    if(getDisplayNumber() != -1 && gadget instanceof Mobile)
    {
        Mobile mobile = (Mobile)item.get(getDisplayNumber()); // Cast the gadget to a Mobile object
        mobile.makeCall(getPhoneNumber(), getDurationMinutes()); // Make a call using the Mobile object
    }
    else
    {
        // Display an error message if the gadget is not a Mobile
        JOptionPane.showMessageDialog(frame, "ERROR!, This is not a mobile");
    }
}
```

Pseudocode:

Define a method called makeCall:

 Retrieve a gadget from the list of items based on a display number

If the display number is valid and the gadget is a mobile:

 Cast the gadget to a Mobile object

 Attempt to make a call using the Mobile object with a phone number and duration

Else:

 Display an error message saying "ERROR!, This is not a mobile"

End of method

3.5. GadgetshopGUI class (public void viewAll()) method and Pseudocode.

This is a method that displays all the gadgets in a list of items. In simple terms, this method goes through each gadget in a list of items and displays its index and details.

```
public void viewAll()
{
    for(Gadget list:item) // Iterate through the list of items
    {
        // Display the index of the gadget in the list
        System.out.print(item.indexOf(list) + 1 + "\n");
        list.display(); // Call the display method of the gadget
    }
}
```

Pseudocode:

Define a method called viewAll:

 For each gadget in the list of items:

 Print the position of the gadget in the list (add 1 because positions usually start from 1, not 0)

 Display the details of the gadget

End of method

3.6. GadgetshopGUI class (public void clear()) method and Pseudocode.

This method is designed to eliminate the text in various text fields of a user interface. It targets multiple textboxes, including modelTextbox, priceTextbox, weightTextbox, sizeTextbox, displayNumberTextbox, phoneNumberTextbox, durationMinutesTextbox, AvailableMemoryTextbox, and creditTextbox. By setting the text in these fields to null, any previous text will be erased.

```
/**
 * Method to clear all text fields in the GUI.
 */
public void clear()
{
    // Set the text of all text fields to null
    modelTextbox.setText(null);
    priceTextbox.setText(null);
    weightTextbox.setText(null);
    sizeTextbox.setText(null);
    displayNumberTextbox.setText(null);
    phoneNumberTextbox.setText(null);
    durationMinutesTextbox.setText(null);
    AvailableMemoryTextbox.setText(null);
    creditTextbox.setText(null);
}
```

Pseudocode:

Define a method called clear:

Set the text of modelTextbox to null

Set the text of priceTextbox to null

Set the text of weightTextbox to null

Set the text of sizeTextbox to null

Set the text of displayNumberTextbox to null

Set the text of phoneNumberTextbox to null

Set the text of durationMinutesTextbox to null

Set the text of AvailableMemoryTextbox to null

Set the text of creditTextbox to null

End of method

3.7. GadgetshopGUI class (public int getCredit()) method and Pseudocode.

This method retrieves the text from a text field named creditTextbox, converts that text into an integer, and then returns this integer. The integer represents the calling credit.

```
/**
 * Method to retrieve the calling credit from the credit text box.
 * Parses the integer value from the credit text box and returns it.
 * @return The calling credit retrieved from the credit text box.
 */
public int getCredit()
{
    int callingCredit = Integer.parseInt(creditTextbox.getText()); // Parse the integer value from the credit text box
    return callingCredit; // Return the calling credit
}
```

Pseudocode:

Define a method called getCredit:

Parse the text of creditTextbox to an integer and assign it to callingCredit

Return callingCredit

End of method

Section 3.8. GadgetshopGUI class (private Double getprice()) method and Pseudocode.

This method retrieves the text from a text field named priceTextbox, converts that text into a double (a decimal number), and then returns this double. The double represents the price.

```
/**
 * Private method to retrieve the price from the price text box.
 * Parses the double value from the price text box and returns it.
 * @return The price retrieved from the price text box.
 */
private double getprice() |
{
    double aPrice = Double.parseDouble(priceTextbox.getText()); // Parse the double value from the price text box
    return aPrice; // Return the price
}
```

Pseudocode:

Define a method called getprice:

Parse the text of priceTextbox to a double and assign it to aPrice

Return aPrice

End of method

3.9. GadgetshopGUI class (private int getweight()) method and Pseudocode.

This method retrieves the text from a text field named weightTextbox, converts that text into an integer, and then returns this integer. The integer represents the weight.

```
/**
 * Private method to retrieve the weight from the weight text box.
 * Parses the integer value from the weight text box and returns it.
 * @return The weight retrieved from the weight text box.
 */
private int getweight()
{
    int aWeight = Integer.parseInt(weightTextbox.getText()); // Parse the integer value from the weight text box
    return aWeight; // Return the weight
}
```

Pseudocode:

Define a method called getweight:

Parse the text of weightTextbox to an integer and assign it to aWeight

Return aWeight

End of method

4. GadgetshopGUI class (private String getsize()) method and Pseudocode.

This method retrieves the text from a text field named sizeTextbox and then returns this text. The text represents the size.

```
/**
 * Private method to retrieve the size from the size text box.
 * Returns the text entered in the size text box.
 * @return The size retrieved from the size text box.
 */
private String getsize()
{
    String aSize = sizeTextbox.getText(); // Retrieve the text entered in the size text box
    return sizeTextbox.getText(); // Return the size
}
```

Pseudocode:

Define a method called getsize:

Retrieve the text of sizeTextbox and assign it to aSize

Return the text of sizeTextbox

End of method

4.1. GadgetshopGUI class (public String getaModel()) method and Pseudocode.

This method retrieves the text from a text field named modelTextbox and then returns this text. The text represents the model.

```
/**
 * Method to retrieve the model from the model text box.
 * Returns the text entered in the model text box.
 * @return The model retrieved from the model text box.
 */
public String getaModel()
{
    String aModel = modelTextbox.getText();// Retrieve the text entered in the model text box
    return aModel;// Return the model
}
```

Pseudocode:

Define a method called getaModel:

Retrieve the text of modelTextbox and assign it to aModel

Return aModel

End of method

4.2. GadgetshopGUI class (public int getAvailableMemory()) method and Pseudocode.

This method retrieves the text from a text field named AvailableMemoryTextbox, converts that text into an integer, and then returns this integer. The integer represents the available memory.

```
/**
 * Method to retrieve the available memory from the AvailableMemory text box.
 * Parses the integer value from the text box and returns it.
 * @return The available memory retrieved from the AvailableMemory text box.
 */
public int getAvailableMemory()
{
    int AvailableMemory = Integer.parseInt(AvailableMemoryTextbox.getText());// Parse the integer value from the Availab
    return AvailableMemory;// Return the available memory
}
```

Pseudocode:

Define a method called getAvailableMemory:

Parse the text of AvailableMemoryTextbox to an integer and assign it to AvailableMemory

Return AvailableMemory

End of method

4.3. GadgetshopGUI class (public int getDisplayNumber ()) method and Pseudocode.

This method retrieves the text from a text field named displayNumberTextbox, tries to convert the text into an integer, and then returns the integer. If the text cannot be converted into an integer (for example, if the text is not a valid number), it catches the NumberFormatException and prints the stack trace of the exception. The integer represents the display number.

```
/**
 * Method to retrieve the display number from the displayNumber text box.
 * Parses the integer value from the text box and returns it.
 * @return The display number retrieved from the displayNumber text box.
 */
public int getDisplayNumber()
{
    int displayNumber = -1; // Initialize the display number
    try
    {
        displayNumber = Integer.parseInt(displayNumberTextbox.getText()); // Parse the integer value from the displayNumber
    }
    catch (NumberFormatException e)
    {
        // Handle the case where the text is not a valid integer
        e.printStackTrace(); // You can log the error or handle it as needed
    }
    return displayNumber; // Return the display number
}
```

Pseudocode:

Define a method called getDisplayNumber:

 Initialize displayNumber to -1

 Try

 Parse the text of displayNumberTextbox to an integer and assign it to displayNumber

 Catch NumberFormatException e

 Handle the case where the text is not a valid integer

 Print the stack trace of the exception

 End Try-Catch

 Return displayNumber

End of method

4.4. GadgetshopGUI class (public int getDurationMinutes ()) method and Pseudocode.

This corrected method now retrieves the text from the durationMinutesTextbox, converts that text into an integer, and then returns this integer. The integer represents the duration in minutes. Please note that if the text in durationMinutesTextbox is not a valid integer, Integer.parseInt will throw a NumberFormatException

```
/**
 * Method to retrieve the duration in minutes from the durationMinutes text box.
 * Parses the integer value from the text box and returns it.
 * @return The duration in minutes retrieved from the durationMinutes text box.
 */
public int getDurationMinutes()
{
    int DurationMinutes = Integer.parseInt(durationMinutesTextbox.getText()); // Parse the integer value from the durationMinutes text box
    return DurationMinutes; // Return the duration in minutes
}
```

Pseudocode:

Define a method called getDurationMinutes:

Parse the text of durationMinutesTextbox to an integer and assign it to DurationMinutes

Return DurationMinutes

End of method

4.5. GadgetshopGUI class (public String getPhoneNumber()) method and Pseudocode.

This method retrieves the text from a text field named phoneNumberTextbox and then returns this text. The text represents the phone number.

```
/**
 * Method to retrieve the phone number from the phoneNumber text box.
 * Returns the text entered in the phoneNumber text box.
 * @return The phone number retrieved from the phoneNumber text box.
 */
public String getPhoneNumber()
{
    String PhoneNumber = phoneNumberTextbox.getText(); // Retrieve the text entered in the phoneNumber text box
    return PhoneNumber; // Return the phone number
}
```

Pseudocode.

Define a method called getPhoneNumber:

Retrieve the text of phoneNumberTextbox and assign it to PhoneNumber

Return PhoneNumber

End of method

4.6. Gadget class method and Pseudocode.

This class represents a generic gadget with four attributes: model, price, weight, and size. The model is a string that represents the model of the gadget. The price is a double that represents the price of the gadget. The weight is an integer that represents the weight of the gadget. The size is a string that represents the size of the gadget.

```
/**
 * The Gadget class represents a generic gadget with model, price, weight, and size attributes.
 */
public class Gadget
{
    private String model;
    private double price;
    private int weight;
    private String size;
}
```

Pseudocode:

Define a class called Gadget:

Declare private string attribute called model

Declare private double attribute called price

Declare private integer attribute called weight

Declare private string attribute called size

End of class

4.7. Gadget class, (public Gadget (String model, double price, int weight, String size)) method and Pseudocode.

This constructor initialises a Gadget object's attributes with provided values for model, price, weight, and size.

```
public Gadget(String model, double price, int weight, String size)
{
    this.model = model;
    this.price = price;
    this.weight = weight;
    this.size = size;
}
```

Pseudocode:

Define a constructor for the Gadget class with parameters model, price, weight, and size:

Set the model attribute of the object to the provided model parameter

Set the price attribute of the object to the provided price parameter

Set the weight attribute of the object to the provided weight parameter

Set the size attribute of the object to the provided size parameter

End of constructor

4.8. Gadget class, (public String getModel()) method and Pseudocode.

This method returns the value of the model attribute of an object of the Gadget class in which this method was defined.

```
/**
 * Accessor method to retrieve the model of the gadget.
 * @return The model of the gadget.
 */
public String getModel()
{
    return model;
}
```

Pseudocode:

Define a method called getModel:

Return the model attribute of the object

End of method

4.9. Gadget class, (public double getPrice()) method and Pseudocode.

This method returns the value of the price attribute of an object of the Gadget class in which this method was defined.

```
/**
 * Accessor method to retrieve the price of the gadget.
 * @return The price of the gadget.
 */
public double getPrice()
{
    return price;
}
```

Pseudocode:

Define a method called getPrice:

Return the price attribute of the object

End of method

5. Gadget class, (public int getWeight()) method and Pseudocode.

This method returns the value of the weight attribute of an object of the Gadget class in which this method was defined.

```
/**
 * Accessor method to retrieve the weight of the gadget.
 * @return The weight of the gadget.
 */
public int getWeight()
{
    return weight;
}
```

Pseudocode:

Define a method called getWeight:

Return the weight attribute of the object

End of method

5.1. Gadget class, (public String getSize()) method and Pseudocode.

This method returns the value of the size attribute of an object of the Gadget class in which this method was defined.

```
/**
 * Accessor method to retrieve the size of the gadget.
 * @return The size of the gadget.
 */
public String getSize()
{
    return size;
}
```

Pseudocode:

Define a method called getSize:

Return the size attribute of the object

End of method

5.2. Gadget class, (public String getSize()) method and Pseudocode.

This method returns the value of the size attribute of an object of the Gadget class in which this method was defined.

```
/**
 * Accessor method to retrieve the size of the gadget.
 * @return The size of the gadget.
 */
public String getSize()
{
    return size;
}
```

Pseudocode:

Define a method called getSize:

Return the size attribute of the object

End of method

5.3. Gadget class, (public void display()) method and Pseudocode.

This method prints by console the attributes of an object of the Gadget class in which this method was defined. Specifically, it prints the model, price, weight, and size attributes.

```
/**
 * Display method to output the details of the gadget.
 */
public void display()
{
    System.out.println("Model: " + model);
    System.out.println("Price: £ " + price);
    System.out.println("Weight: " + weight + "grams");
    System.out.println("Size: " + size);
}
```

Pseudocode:

Define a method called display:

Print "Model: " followed by the model attribute of the object

Print "Price: £ " followed by the price attribute of the object

Print "Weight: " followed by the weight attribute of the object and "grams"

Print "Size: " followed by the size attribute of the object

End of method

5.4. Mobile class, (public Mobile(String model, double price, int weight, String size, int callingCredit)) method and Pseudocode.

This is a constructor for Mobile class. This constructor takes five parameters: model, price, weight, size, and callingCredit. It calls the constructor of the superclass Gadget with the first four parameters to initialise the inherited attributes, and then it initializes the callingCredit attribute of the Mobile class with the value provided.

```
public Mobile(String model, double price, int weight, String size, int callingCredit)
{
    super(model, price, weight, size);
    this.callingCredit = callingCredit;
}
```

Pseudocode:

Define a constructor for the Mobile class with parameters model, price, weight, size, and callingCredit:

Call the constructor of the superclass Gadget with parameters model, price, weight, and size.

Set the callingCredit attribute of the object to the provided callingCredit parameter.

End of constructor.

5.5. Mobile class, (public int getCallingCredit()) method and Pseudocode.

This method returns the value of the callingCredit attribute of an object of the Mobile class, in which this method was defined. The callingCredit represents the calling credit of the mobile phone.

```
public int getCallingCredit()
{
    return callingCredit;
}
```

Pseudocode:

Define a method called getCallingCredit:

Return the callingCredit attribute of the object

End of method

5.6. Mobile class, (public void addCallingCredit(int creditToAdd)) method and Pseudocode.

This method requires an integer parameter called creditToAdd. The method is defined within the Mobile class object. If the creditToAdd parameter is a positive integer value, it will be added to the callingCredit attribute of the Mobile class object. However, if the creditToAdd parameter is not a positive integer value, the method will print a message asking the user to enter a positive amount.

```
public void addCallingCredit(int creditToAdd)
{
    if (creditToAdd > 0) // Check if credit to add is positive
    {
        this.callingCredit += creditToAdd; // Add credit
    }
    else
    {
        System.out.println("Please enter a positive amount for calling credit."); // Prompt user for positive amount
    }
}
```

Pseudocode: Define a method called addCallingCredit with parameter creditToAdd:

```

    If creditToAdd is greater than 0
        Add creditToAdd to the callingCredit attribute of the object
    Else
        Print "Please enter a positive amount for calling credit."
    End If
End of method
```

5.7. Mobile class, (public void makeCall(String phoneNumber, int durationMinutes)) method and Pseudocode.

This particular method requires two parameters: a phoneNumber and durationMinutes. Its primary function is to check whether the callingCredit attribute of the object is greater than or equal to the durationMinutes parameter. If the credit is enough, it will notify that a call is being made to the provided phoneNumber for the specified duration, otherwise, it will indicate that there is insufficient credit to make the call.

```
public void makeCall(String phoneNumber, int durationMinutes)
{
    if (callingCredit >= durationMinutes) // Check if enough credit for the call
    {
        System.out.println("Calling " + phoneNumber + " for " + durationMinutes + " minutes :"); // If is true make the call
    }
    else
    {
        System.out.println("Insufficient credit to make the call."); // If is false display insufficient credit message
    }
}
```

Pseudocode: Define a method called makeCall with parameters phoneNumber and durationMinutes:

```

    If callingCredit is greater than or equal to durationMinutes
        Print "Calling " followed by phoneNumber, " for ", durationMinutes, " minutes :"
    Else
        Print "Insufficient credit to make the call."
    End If
End of method
```


5.8. Mobile class, (public void display()) method and Pseudocode.

In this context, display() is a subclass instance method. super.display() calls the superclass method to output gadget details. The line System.out.println("Calling credit: " + callingCredit + " minutes:") outputs the calling credit in minutes, callingCredit is an instance variable of the subclass.

```
/**
 * Display method to output the details of the mobile phone.
 */
public void display()
{
    super.display();// Call display method of superclass to output gadget details
    System.out.println("Calling credit: " + callingCredit + " minutes:");// Output calling credit
}
```

Pseudocode:

Method display

Call display method of superclass

Output "Calling credit: " concatenated with the value of callingCredit concatenated with " minutes"

End Method

5.9. MP3 class, (public MP3(String model, double price, int weight, String size, double availableMemory)) method and Pseudocode.

The MP3 method is a constructor of a subclass that calls the constructor of the superclass with provided parameters using the 'super' keyword. It also assigns the availableMemory of the MP3 player with the provided value using the 'this' keyword.

```
public MP3(String model, double price, int weight, String size, double availableMemory)
{
    // Calling the superclass constructor with the provided model, price, weight, and size
    super(model, price, weight, size);

    // Assigning the available memory of the MP3 player with the provided value.
    this.availableMemory = availableMemory;
}
```

Pseudocode:

Constructor MP3 takes parameters model, price, weight, size, availableMemory

Call superclass constructor with parameters model, price, weight, size

Assign availableMemory of this instance to the provided availableMemory value

End Constructor

6. MP3 class, (public double getAvailableMemory()) method and Pseudocode.

This is a method that retrieves the availableMemory of the MP3 player. Getter methods are typically used to access instance variable values.

```
public double getAvailableMemory()  
{  
    return availableMemory; // Return the available memory of the MP3 player.  
}
```

Pseudocode:

Method getAvailableMemory

Return the value of availableMemory

End Method

6.1. MP3 class, (public void downloadMusic(double memoryUsed)) method and Pseudocode.

The downloadMusic() method downloads music and updates the MP3 player's available memory. It checks if there is enough memory, reduces the available memory if there is, and displays a message with the remaining memory. If there is not enough memory, it displays an error.

```
public void downloadMusic(double memoryUsed)  
{  
    if (memoryUsed > 0) // Check if there is enough available memory to download the music.  
    {  
        availableMemory -= memoryUsed; // If there is enough memory, reduce the memory used for the downloaded music.  
        // Display a message indicating successful download and the remaining available memory.  
        System.out.println("Music download successfully! Available memory: " + availableMemory + "MB");  
    }  
    else  
    {  
        // If there is not enough memory, display an error message.  
        System.out.println("Sorry, there is not enough memory to download the music.");  
    }  
}
```

Pseudocode:

Method downloadMusic takes parameter memoryUsed

If memoryUsed is greater than 0

Subtract memoryUsed from availableMemory

Output "Music download successfully! Available memory: " concatenated with the value of availableMemory concatenated with "MB"

Else

Output "Sorry, there is not enough memory to download the music."

End If

End Method

6.2. MP3 class, (public void display()) method and Pseudocode.

The display() method is an instance method of a subclass. super.display() calls the method of the superclass. System.out.println() outputs availableMemory in MB.

```
public void display()
{
    super.display();// Call the superclass display method to display basic information.

    // Display the available memory of the MP3 player.
    System.out.println("available Memory: " +availableMemory+ "MB: ");
}
```

Pseudocode:

Method display

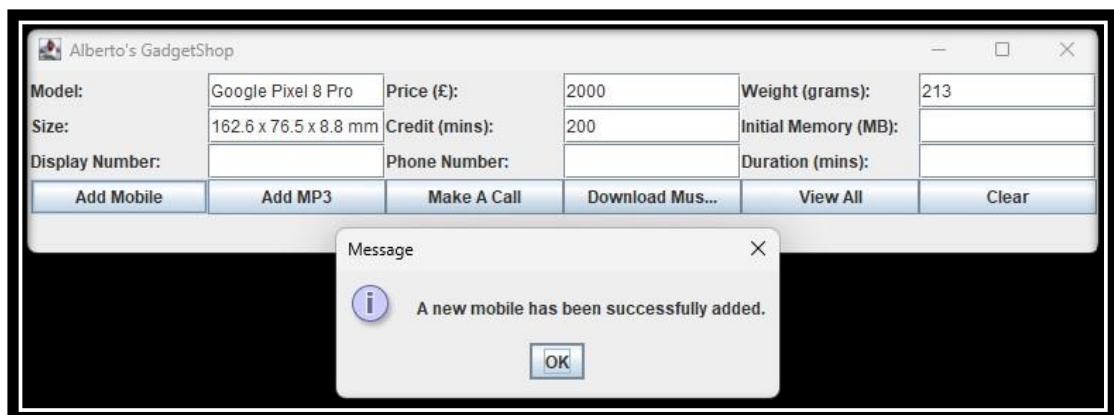
Call display method of superclass

Output "available Memory: " concatenated with the value of availableMemory concatenated with "MB"

End Method

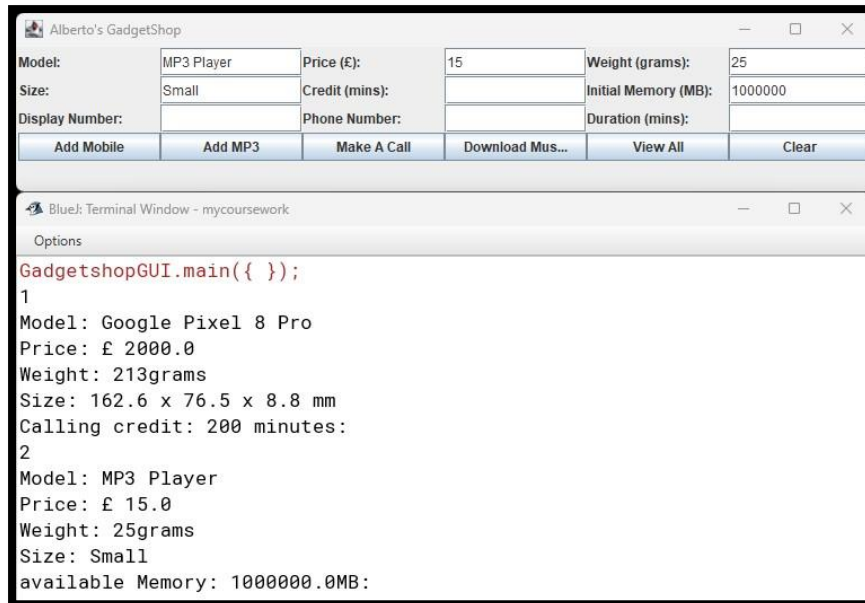
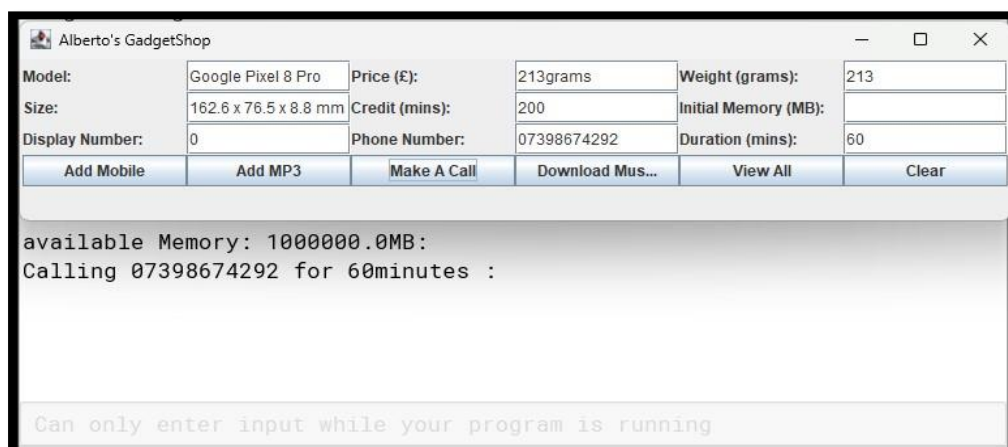
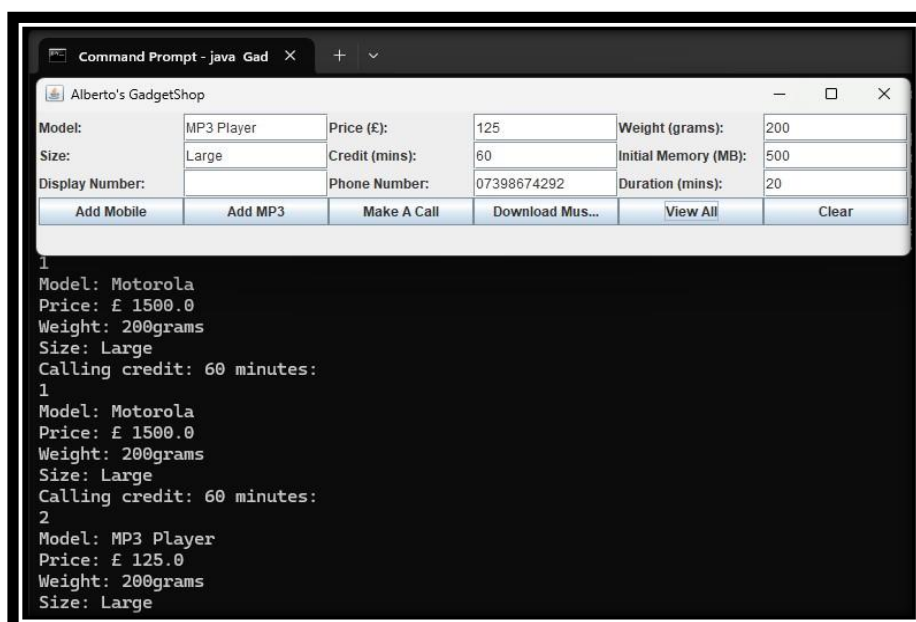
Section 7. Testing the functionality of the program.

Test 1: Adding a mobile to the array list.

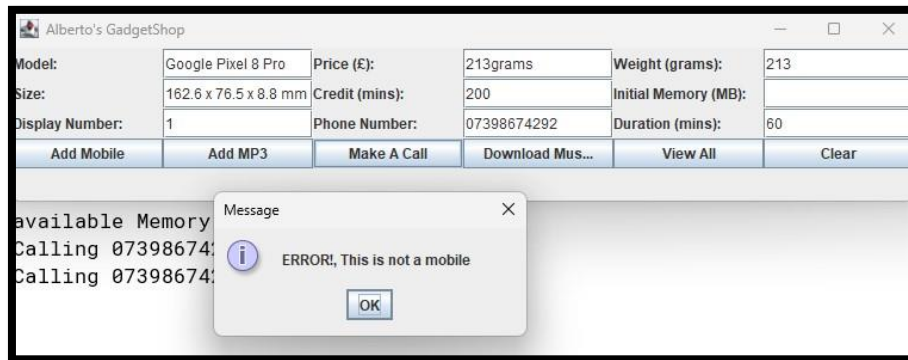


Test 2: Adding an MP3 Player to the array list.



Test 3: Displaying all gadgets stored in the array list.**Test 4:** Making a Call.**Test 5:** Compiling and running the program from the command prompt.

Test 6: Appropriate dialogue appears when unsuitable value is entered.

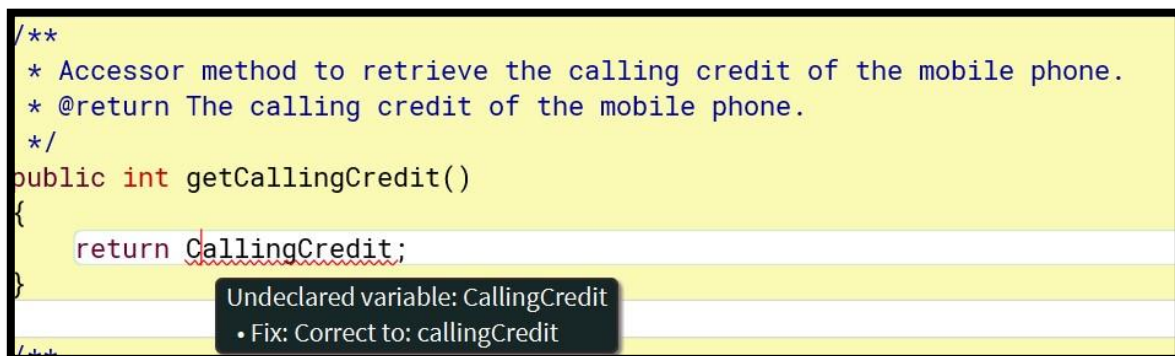


Test 7: An appropriate message appears when attempting to make a call with insufficient credit.



Section 8. Error detection and error correction.

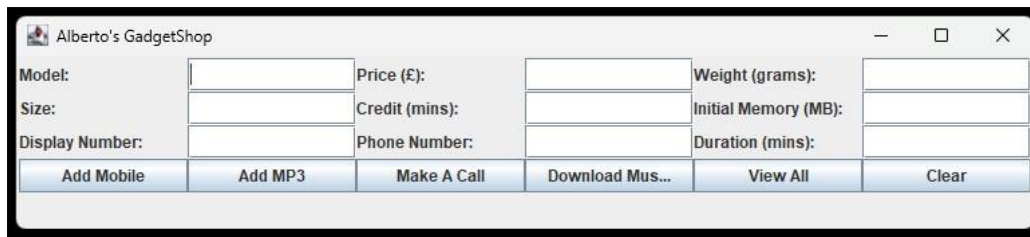
Case 8.1. : One of the most significant mistakes I made during the project was a syntax error. Specifically, when creating a getter method for "**getCallingCredit**," I mistakenly used a capital "C" after "**get**." Please refer to the image below for a clearer understanding of the syntax error and how I resolved it.



Case 8.2. : After completing my GUI, I tested it by compiling it. However, I was not satisfied with the final outcome, as my text fields and buttons appeared untidy. I searched on the internet for a solution and found a line of code: `contentPane.add (Box.createHorizontalGlue());` This code adds a flexible space, also known as "glue," which can expand as required to fill any extra horizontal space in the layout. If components occupy all the available space and no additional space is left, the glue will have a width of zero.



Before I added more horizontal space.

A screenshot of a Java Swing window titled "Alberto's GadgetShop". The window contains a form with three rows of labels and text boxes: "Model:", "Price (£):", "Weight (grams):" in the first row; "Size:", "Credit (mins):", "Initial Memory (MB):" in the second row; and "Display Number:", "Phone Number:", "Duration (mins):" in the third row. Below the text boxes is a row of six buttons: "Add Mobile", "Add MP3", "Make A Call", "Download Mus...", "View All", and "Clear". The window has standard OS window controls (minimize, maximize, close) in the top right corner.

After more horizontal space was added.

Case 8.3. : I am having trouble with the download music function. For some reason, it is not working properly. There are no syntax errors; the logic in my method seems to be correct, but when I click on the button, nothing happens. Additionally, there is no output displayed in the terminal. I have attempted to troubleshoot the issue on my own, but I have been unsuccessful thus far."

Conclusion.

I have found this journey to be incredibly enriching. After putting so much effort into this project, I have learned many things, such as how to build a GUI in Java, how buttons and text boxes work, and the process behind a button and key when they are pressed. I have also learned how to use an array list, how to declare variables wisely, the importance of a constructor method, and the implementation of Boolean operators and try and catch statements, how inheritances classes work and the importance of redacting a good report.

I have spent a lot of time and faced many adversities to get to this point, including the recent passing of my beloved mother, which has hit me extremely hard. Nevertheless, I am proud of myself. Although the project may not have turned out perfectly, I know that if I continue practising and working hard on my coding skills and problem-solving abilities, I can become a better programmer."