

**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: 22030807**



**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:) SERGIU MITA

Date: 01 MAY 2024

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 .....

(i) **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.**

(ii) **Falsifying data in experimental results.**

(iii) 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.

(iv) **Bribery or attempted bribery of a person thought to have some influence on the candidate's assessment.**

(v) Collusion to present joint work as the work solely of one individual.

(vi) Plagiarism, where the work or ideas of another are presented as the candidate's own.

(vii) Other conduct calculated to secure an advantage on assessment.

(viii) Assisting in any of the above.

Some notes on what this means for students:

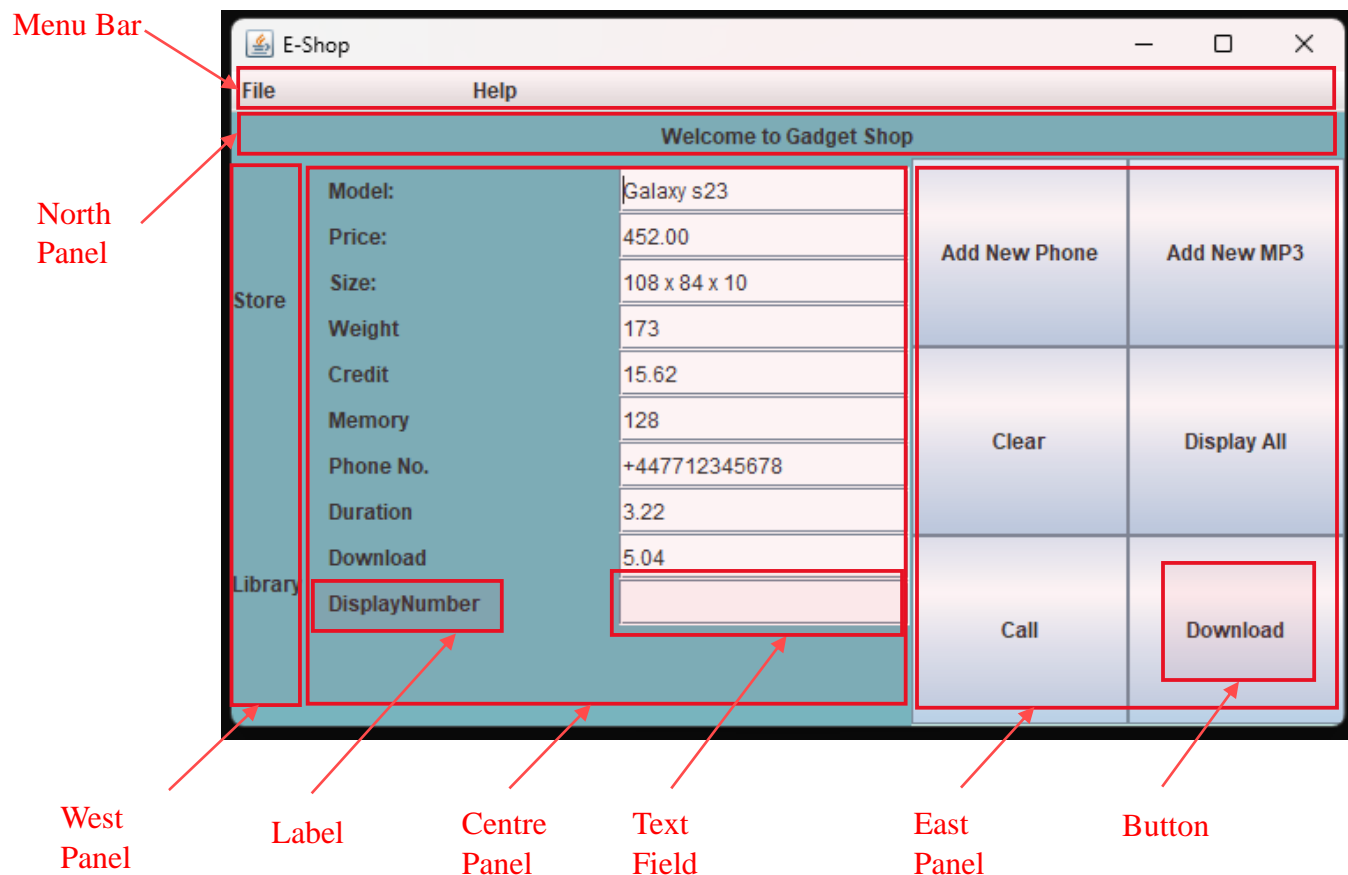
1. Copying another student's work is an offence, whether from a copy on paper or from a computer file, and in whatever form the intellectual property being copied takes, including text, mathematical notation and computer programs.

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 = mc<sup>2</sup> (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.

## Table of Contents

1	Introduction and explanation about the application .....	P3
2	The GitHub link to the application .....	P3
3	Class UML diagram of the Project .....	P4
4	Pseudocode for the following button-handling methods .....	P5
5	Overview of the remaining methods in the project .....	P14
6	The evidence of the following testing .....	P19
7	Errors .....	P31
8	Conclusion .....	P35
9	Appendix .....	P36
10	References .....	P53

## 1. Introduction and explanation about the application

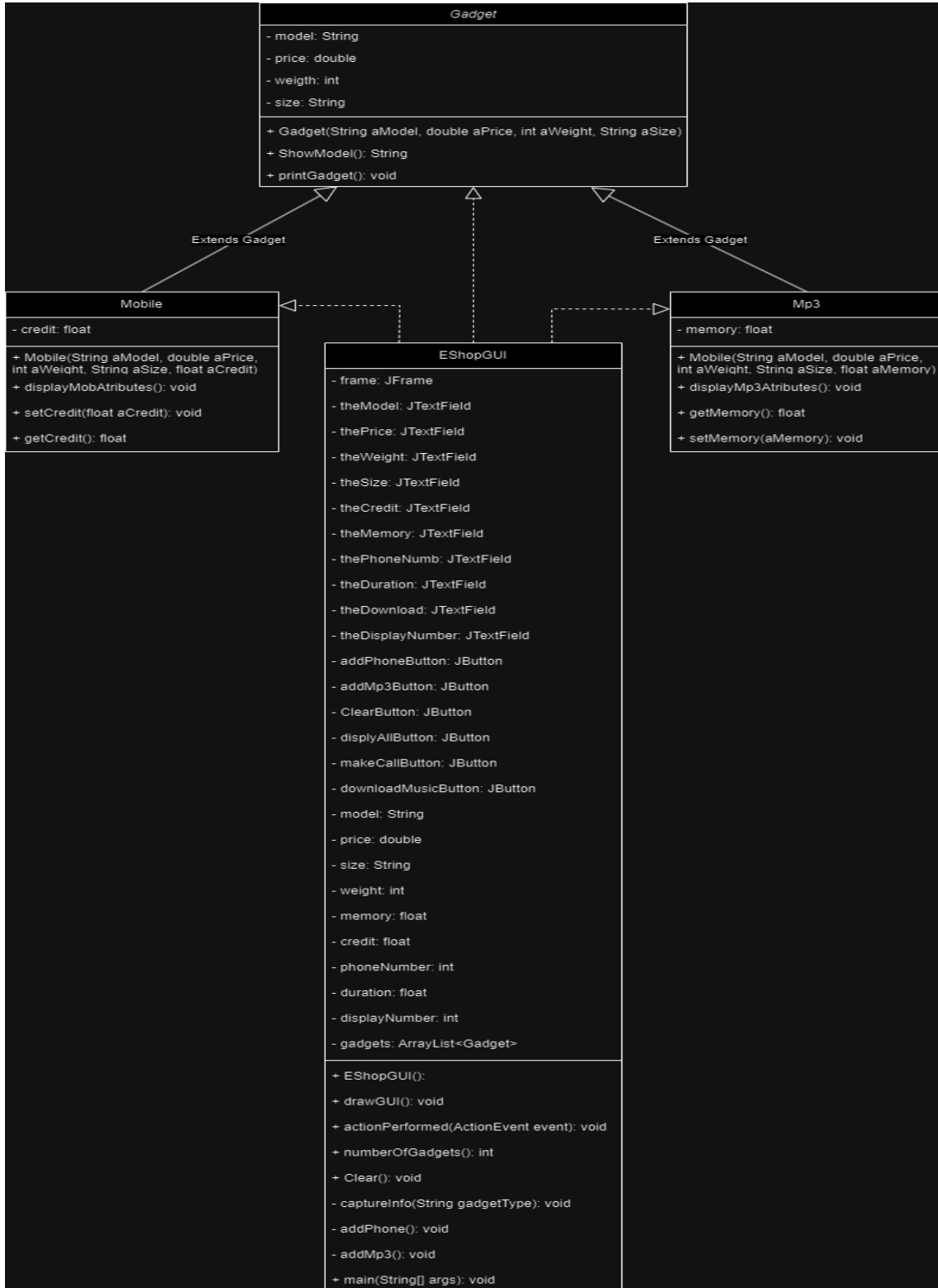


This is the Graphical User Interface which is composed from a windows frame called E-Shop. The frame is compound from 2 elements Menu Bar and Content Pane. The content pane was set a Border Layout which contains 4 panels (North, West, Centre, and East panels). In the North and West panels, we can observe only labels in the Centre panel we can see labels and Text fields which will take the input information from the user and in the East panel we can find 6 buttons which are doing some actions when they are pressed.

## 2. The GitHub link to the application.

<https://github.com/sergiumi/Corswork-report-GadgetShop.git>

### 3. Class UML diagram of the Project.



The UML diagram is a schematic representation of the program, which includes classes and the relationships between them. Each class holds instances and methods. The E-Shop is structured with two subclasses: the Mobile class and the Mp3 class, both of which extend the superclass called Gadget class. Additionally, there is the EShopGUI class, which encapsulates the GUI elements, and the ArrayList. Finally, there is the Main class, which allows us to start the program from the command prompt of the machine.

## 4. Pseudocode for the following button-handling methods.

### 1) Adding a Mobile and Mp3 method

```
private void captureInfo(String gadgetType) // capture method which checks if every text field is completed correctly
//and it is using a parameter to check which gadget to add to gadgetList
{
    try //using try statement to make the program more error stable and user friendly
    {
        model = theModel.getText(); //read the text field from GUI and assigning it to a variable
        if (model.isEmpty()) //Check if the text field is empty
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The MODEL field is empty. Please insert text.");
            return; //break the program and return to the invoked method
        }

        price = Double.parseDouble(thePrice.getText()); //read the text field from GUI,
        //convert it from string to double and assigning it to variable price
        if (price <= 0) //check if the price is less then 0
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The PRICE field is empty or invalid. Please insert a positive number.");
            return; //break the program and return to the invoked method
        }

        size = theSize.getText(); //read the text field from GUI and assigning it to variable size
        if (size.isEmpty()) //Check if the text field is empty
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The SIZE field is empty. Please insert text.");
            return; //break the program and return to the invoked method
        }

        weight = Integer.parseInt(theWeight.getText()); //read the text field from GUI,
        //convert it from string to integer type and assigning it to variable weight
        if (weight <= 0) //check if the price is less then 0
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The WEIGHT field is empty or invalid. Please insert a positive number.");
            return; //break the program and return to the invoked method
        }
    }
}
```

These 5 images presents the same mothod which is a nested method writed in java language and its pseudocode which looks quiet the same. This method involves “try” block in the method to check if any errors occurd during the execution of the program without stopping the program and be more friendly for the user. I used this bloc in the method to check if all the text fields are completed as I requered and after the check is completed to add a device by clicking the add phone or add Mp3 button as well in this method I used a parameter to add the gadget to the array list.

```

if (gadgetType.equals("Phone")) //checking gadget type if it is a phone
{
    //add a new attribute if the gadget it's a phone
    credit = Float.parseFloat(theCredit.getText()); //read the text field from GUI,
    //convert it from string to float type and assigning it to variable credit
    if (credit <= 0) //check if the credit is less than 0
    {
        //pop-up an informative message for user
        JOptionPane.showMessageDialog(frame, "The CREDIT field is empty or invalid. Please insert a positive number.");
        return; //break the program and return to the invoked method
    }
    addPhone(); //using addPhone method to add new phone to gadget list
    Clear(); //using clear method to clear the fields
}
else if (gadgetType.equals("Mp3")) //if it is an mp3 the program is executing the next code
{
    //add a new attribute if the gadget it's an mp3
    memory = Float.parseFloat(theMemory.getText()); //read the text field from GUI,
    //convert it from string to float type and assigning it to variable credit
    if (memory <= 0) //check if the memory is less than 0
    {
        //pop-up an informative message for user
        JOptionPane.showMessageDialog(frame, "The MEMORY field is empty or invalid. Please insert a positive number.");
        return; //break the program and return to the invoked method
    }
    addMp3(); //using addMp3 method to add new mp3 to gadget list
    Clear(); //using clear method to clear the fields
}
}

catch (NumberFormatException e) //catch the error without stopping the program
{
    //pop-up an informative message for user
    JOptionPane.showMessageDialog(frame, "An error occurred while processing the input.");
}
}

```

```

Method: captureInfo(gadgetType)
function captureInfo(gadgetType):
    try:
        model = theModel.getText()
        if model is empty:
            showErrorMessage("The MODEL field is empty. Please insert text.")
            return

        price = parseDouble(thePrice.getText())
        if price <= 0:
            showErrorMessage("The PRICE field is empty or invalid. Please insert a positive number.")
            return

        size = theSize.getText()
        if size is empty:
            showErrorMessage("The SIZE field is empty. Please insert text.")
            return

        weight = parseInt(theWeight.getText())
        if weight <= 0:
            showErrorMessage("The WEIGHT field is empty or invalid. Please insert a positive number.")
            return

        if gadgetType equals "Phone":
            credit = parseFloat(theCredit.getText())
            if credit <= 0:
                showErrorMessage("The CREDIT field is empty or invalid. Please insert a positive number.")
                return
            addPhone()
            Clear()
        else if gadgetType equals "Mp3":
            memory = parseFloat(theMemory.getText())
            if memory <= 0:
                showErrorMessage("The MEMORY field is empty or invalid. Please insert a positive number.")
                return
            addMp3()
            Clear()

    catch(NumberFormatException e):
        showErrorMessage("An error occurred while processing the input.")

```

```
private void addPhone()
{
    //the information to create a new gadget
    Gadget newGadget = new Mobile(model, price, weight, size, credit);
    //add the newly created Gadget object to the gadgets ArrayList
    gadgets.add(newGadget);
    System.out.println("+1 Phone added");
    System.out.println("Model: " + model);
    System.out.println("-----");
    System.out.println();
}
```

```
private void addMp3()
{
    //the information to create a new gadget
    Gadget newGadget = new Mp3(model, price, weight, size, memory);
    //add the newly created Gadget object to the gadgets ArrayList
    gadgets.add(newGadget);
    System.out.println("+1 MP3 added");
    System.out.println("Model: " + model);
    System.out.println("-----");
    System.out.println();
}
```

```
function addPhone():
    // Create a new Mobile object with the provided information
    newGadget = create Mobile object with model, price, weight, size, and credit

    // Add the newly created Mobile object to the gadgets ArrayList
    add newGadget to gadgets ArrayList

    // Output a confirmation message
    output "+1 Phone added"
    output "Model: " + model
    output "-----"

function addMp3():
    // Create a new Mp3 object with the provided information
    newGadget = create Mp3 object with model, price, weight, size, and memory

    // Add the newly created Mp3 object to the gadgets ArrayList
    add newGadget to gadgets ArrayList

    // Output a confirmation message
    output "+1 MP3 added"
    output "Model: " + model
    output "-----"
    output newline
```

In the last two images, you can observe the proper methods for adding a phone and an MP3 player to the gadget list. These methods involve creating a new gadget object in the array list using the parameters that were declared as global variables. Additionally, they display messages in the BlueJ terminal for confirmation.

## 2) Clear the fields

```
public void Clear() // clear the fields method
{
    // set new values to each field in the GUI
    theModel.setText("");
    thePrice.setText("");
    theWeight.setText("");
    theSize.setText("");
    theCredit.setText("");
    theMemory.setText("");
    thePhoneNumb.setText("");
    theDuration.setText("");
    Download.setText("");
    DisplayNumber.setText("");
}
```

```
function Clear():
    // Set empty values to each field in the GUI
    setText(theModel, "")
    setText(thePrice, "")
    setText(theWeight, "")
    setText(theSize, "")
    setText(theCredit, "")
    setText(theMemory, "")
    setText(thePhoneNumb, "")
    setText(theDuration, "")
    setText(Download, "")
    setText(DisplayNumber, "")
```

The Clear method is employed to reset all text fields within the GUI interface without altering any other aspects of the program. It is doing this by replacing the current text within each field with an empty string."



### 3) Displaying all gadgets in the array list

```
if (command.equals("Display All")) // add if statement to check if "Display All" button was pressed
{
    System.out.println(numberOfGadgets() + " Gadgets in library"); // print statement which
    //shows how many gadgets are in arrayList
    System.out.println(); // empty print statement
    System.out.println("The list of gadgets is : " + gadgets ); // print statement with list of the gadgets
    System.out.println();
    for (Gadget g : gadgets) //Looping through the gadget list
    {
        Gadget currentGad = g; // assigning the current gadget to a variable
        if (currentGad instanceof Mobile) // checking if current gadget belongs to Mobile class
        {
            Mobile mobile = (Mobile) currentGad; // casting the current gadget to Mobile
            //so we can access the mobile methods from the gui class
            mobile.displayMobAttributes(); // calling displayMobAttributes method
            //for the mobile item in arrayList through casting from mobile class to display mobiles attributes
        }
        if (currentGad instanceof Mp3) // checking if current gadget belongs to Mp3 class
        {
            Mp3 mp3 = (Mp3) currentGad; // casting the current gadget to Mp3
            mp3.displayMp3Attributes(); // calling displayMp3Attributes method
        }
    }
}
```

```
public int numberOfGadgets() // number of gadgets method
{
    return gadgets.size(); //returning the size of the arrayList by using the
    //built-in method on lists .size()
}
```

The displayAll method is designed to showcase all the gadgets stored in the array list. The proper code you can spot above and the pseudocode on the next page. By utilizing casting method, I am able to access the specific information stored within the Mobile and Mp3 subclasses directly from the GUI class. The method then proceeds to print this information in the BlueJ terminal."

```

if (command.equals("Display All")) {
    // Print the number of gadgets in the library
    print(numberOfGadgets() + " Gadgets in library");

    // Print an empty line
    print();

    // Print the list of gadgets
    print("The list of gadgets is: " + gadgets);

    // Print an empty line
    print();

    // Loop through the gadgets list
    for each gadget in gadgets {
        // Assign the current gadget to a variable
        currentGadget = gadget;

        // Check if the current gadget is an instance of Mobile
        if (currentGadget is instance of Mobile) {
            // Cast the current gadget to Mobile
            mobile = (Mobile) currentGadget;

            // Call the displayMobAtributes method for the mobile item in the arrayList
            mobile.displayMobAtributes();
        }

        // Check if the current gadget is an instance of Mp3
        if (currentGadget is instance of Mp3) {
            // Cast the current gadget to Mp3
            mp3 = (Mp3) currentGadget;

            // Call the displayMp3Atributes method
            mp3.displayMp3Atributes();
        }
    }
}

```

## 4) Getting the display number from the GUI and Making a call

```
private void checkDuration() // creating checkDuration method to select the mobile from the list and make the call
// the number and to check if that mobile has enough credit to make the call
{
    try //using try method to make this function user friendly and not brake the program
    {
        phoneNumber = Integer.parseInt(thePhoneNumb.getText()); // convert the text from the GUI to type integer and assign to a global variable
        displayNumber = Integer.parseInt(DisplayNumber.getText()); // convert the text from the GUI to type integer and assign to a global variable
        if (displayNumber >= 0 && displayNumber < gadgets.size()) //add condition to check if the display number was entered correctly
        {
            Gadget gadget = gadgets.get(displayNumber); //searching for the gadget index in the list
            if (gadget instanceof Mobile) // checking if the gadget of that index belongs to mobile devices
            {
                Mobile mobile = (Mobile) gadget; // assigning mobile of type gadget to mobile of type mobile
                duration = Float.parseFloat(theDuration.getText()); // convert the text from the GUI to type float and assign to a global variable
                float currCredit = mobile.getCredit(); // creating a new local variable and assign to the method get credit from mobile class
                if (duration >= 0 && duration <= currCredit) // checking if the duration is less than the existing credit of the chosen phone from the gadget list
                {
                    System.out.println("Your initial credit on the phone " + model + " is: " + currCredit); // printing the initial credit
                    currCredit -= duration; //subtract duration from initial credit
                    mobile.setCredit(currCredit); // assign new value after the subtraction
                    System.out.println("Making a call from model " + mobile.showModel() + " to number: " + phoneNumber); // print a message that we are calling a chosen number
                    System.out.println("...");
                    System.out.println("...");
                    System.out.println("...");
                    System.out.println("Your credit on the phone " + model + " is now: " + currCredit); // display the remaining credit
                    System.out.println("-----");
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Try another device or reduce the calling time!"); // popping a message if its not enough credit
                }
            }
            else
            {
                System.out.println("The gadget at index " + displayNumber + " is not a mobile device."); // popping a message if chosen device is not an mobile phone
            }
        }
        else
        {
            JOptionPane.showMessageDialog(frame, "Invalid mobile index."); // popping a message if the index device is not in the list
        }
    }
    catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(frame, "An error occurred while processing the input."); // popping a message if the text field is empty
    }
}
```

```
function checkDuration():
    try:
        phoneNumber = convert to int (thePhoneNumb.getText())
        displayNumber = convert to int (DisplayNumber.getText())

        if displayNumber >= 0 and displayNumber < gadgets.size():
            gadget = gadgets.get(displayNumber)
            if gadget is instance of Mobile:
                mobile = cast gadget to Mobile
                duration = convert to float(theDuration.getText())
                currCredit = mobile.getCredit()
                if duration >= 0 and duration <= currCredit:
                    print("Your initial credit on the phone " + model + " is: " + currCredit)
                    currCredit -= duration
                    mobile.setCredit(currCredit)
                    print("Making a call from model " + mobile.showModel() + " to number: " + phoneNumber)
                    print("...")
                    print("...")
                    print("...")
                    print("Your credit on the phone " + model + " is now: " + currCredit)
                    print("-----")
                else:
                    showErrorMessage("Try another device or reduce the calling time!")
            else:
                print("The gadget at index " + displayNumber + " is not a mobile device.")
        else:
            showErrorMessage("Invalid mobile index.")
    catch(NumberFormatException e):
        showErrorMessage("An error occurred while processing the input.")
```

In the two images above, you can see the initial Java code and the corresponding pseudocode for the function named `checkDuration`. This function encompasses three distinct tasks triggered when the call button is pressed: **Display Number Validation**: This function checks if the display number is entered correctly and verifies if the gadget index associated with the display number corresponds to a mobile device. **Duration and Credit Verification**: It ensures that the duration of the call is less than the available credit on the selected phone. And **Phone Number Validation**: Lastly, it checks if the entered phone number is an integer type.

## 5) Downloading music

```
private void checkMemory() // creating CehckMemory method to select: the Mp3 from the list and download if that Mp3 has enough memory to download music
{
    try //using try method to make this function user friendly and not brake the program
    {
        displayNumber = Integer.parseInt(DisplayNumber.getText()); // convert the text from the GUI to type integer and assign to a global variable
        if (displayNumber >= 0 && displayNumber < gadgets.size()) //add condition to check if the dispaly number was entered correctly
        {
            Gadget gadget = gadgets.get(displayNumber); //serchig for the gadget index in the list

            if (gadget instanceof Mp3)
            {
                Mp3 Mp3 = (Mp3) gadget;
                download = Float.parseFloat(Download.getText());
                float currMemory = Mp3.getMemory();
                if (download >= 0 && download <= currMemory)
                {
                    System.out.println("Your available memory on the Mp3 " + model + " is: " + currMemory); // printing the initial memory
                    currMemory -= download; //subtract download memory from initial memory
                    Mp3.setMemory(currMemory); // assign new value after the subtraction
                    System.out.println("Download music on " + Mp3.showModel()); // print a message that we are downloading music
                    System.out.println("...");
                    System.out.println("Remaning memory on the Mp3 " + model + " is now: " + currMemory ); // display the remaining memory
                    System.out.println("-----");
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Try another device or reduce the download memory!"); // popping a message if its not enoth memory
                }
            }
            else
            {
                System.out.println("The gadget at index " + displayNumber + " is not a Mp3 device."); // popping a message if chosen device is not an Mp3
                System.out.println("-----");
                System.out.println();
            }
        }
        else
        {
            JOptionPane.showMessageDialog(frame, "Invalid Mp3 index."); // popping a message if the index device is not in the list
        }
    }
    catch (NumberFormatException e)
    {
        JOptionPane.showMessageDialog(frame, "An error occurred while processing the input."); // // popping a message if the text field is empty
    }
}
```

The Download method closely resembles the Call method as it involves assigning the display number to an index within the gadget list. It verifies whether the gadget at that index is an MP3. If affirmative, the method proceeds to assess if the MP3 possesses sufficient memory for downloading music, showcasing the remaining memory post-download. The pseudocode of this method can be examined lower.

```

function checkMemory():
    try:
        // Convert text from GUI to integer and assign to a variable
        displayNumber = Integer.parseInt(DisplayNumber.getText())

        // Check if display number is within valid range
        if (displayNumber >= 0 && displayNumber < gadgets.size()):
            // Retrieve gadget at the specified index
            gadget = gadgets.get(displayNumber)

            // Check if the gadget is an instance of Mp3
            if (gadget instanceof Mp3):
                // Cast gadget to Mp3
                Mp3 = (Mp3) gadget

                // Convert text from GUI to float and assign to download variable
                download = Float.parseFloat(Download.getText())

                // Get current memory of the Mp3
                currMemory = Mp3.getMemory()

                // Check if download memory is within available memory
                if (download >= 0 && download <= currMemory):
                    // Print initial memory
                    print("Your available memory on the Mp3 " + model + " is: " + currMemory)

                    // Subtract download memory from initial memory
                    currMemory -= download

                    // Update memory of Mp3
                    Mp3.setMemory(currMemory)

                    // Print message indicating music download
                    print("Download music on " + Mp3.showModel())
                    print("...")

                    // Print remaining memory
                    print("Remaining memory on the Mp3 " + model + " is now: " + currMemory)
                    print("-----")
                else:
                    // Display message if not enough memory for download
                    JOptionPane.showMessageDialog(frame, "Try another device or reduce the download memory!")
            else:
                // Display message if gadget is not an Mp3 device
                print("The gadget at index " + displayNumber + " is not an Mp3 device.")
                print("-----")
                print()
        else:
            // Display message if invalid Mp3 index
            JOptionPane.showMessageDialog(frame, "Invalid Mp3 index.")
    catch NumberFormatException:
        // Display message if an error occurred while processing the input
        JOptionPane.showMessageDialog(frame, "An error occurred while processing the input.")

```

## 5. Overview of the remaining methods in the project.

The remaining methods either contribute directly to the main functionalities within the main methods or are designed in separate classes to enhance program readability. Some of these methods may serve as constructors for other classes, ensuring proper initialization and organization within the program structure.

### a) Remaining methods in EShopGUI() class:

#### I. Constructor method of the EShopGUI class

```
public EShopGUI()
{
    gadgets = new ArrayList<Gadget>(); // creating the gadget array list
    drawGUI(); // calling the draw GUI to set the window for the GUI
}
```

#### II. Here is the drawGUI() method which includes the creation of frames and panels necessary for displaying the Graphical User Interface (GUI)."

```
public void drawGUI()
{
    frame = new JFrame("E-Shop"); //create the frame window
    frame.setLocation(400,100); //set the location of the frame on the screen

    JMenuBar menuBar = new JMenuBar(); //Create new empty menu bar
    frame.setJMenuBar(menuBar); //set the menu bar to the frame

    //File block
    JMenu fileMenu = new JMenu("File"); // create the file menu
    menuBar.add(fileMenu); // add the file menu to the menu bar

    JMenuItem newFileItem = new JMenuItem("New File"); //create a new item
    newFileItem.addActionListener(this); //add the "new file" item to action listener to provide the comand
    fileMenu.add(newFileItem); // add the "new file" item to the file in menu bar menu

    JMenuItem openFileItem = new JMenuItem("Open");
    openFileItem.addActionListener(this);
    fileMenu.add(openFileItem);

    menuBar.add(Box.createHorizontalStrut(100)); // add a space between the 2 items in menu bar
    //menuBar.add(Box.createHorizontalGlue()); // add maximum available space between 2 menu items

    //Help block
    JMenu helpMenu = new JMenu("Help"); // create the file menu
    menuBar.add(helpMenu); // add the file menu to the menu bar

    Container contentPane = frame.getContentPane();//create the spase to be filed with butons and fields

    BorderLayout borderLayout = new BorderLayout(); //create new border layout
    contentPane.setLayout(borderLayout); //set the border layout for the content panel

    //Panel for the North block
    JPanel northPanel = new JPanel(); // crete a new panel
    contentPane.add(northPanel,BorderLayout.NORTH); // add the north border to the top of the window panel

    northPanel.add(new JLabel("Welcome to Gadget Shop")); // add a label to north borer
    northPanel.setBackground(new Color(120,180,190)); //set the color to the north panel

    //Panel for the West block
    JPanel westPanel = new JPanel(); //create a new panel in the left side of the window
    contentPane.add(westPanel,BorderLayout.WEST); // add the west panel to the border layout
    GridLayout westGridLayout = new GridLayout(2,1); //create grid layout in the west bordr of the layout
    westPanel.setLayout(westGridLayout); //setting the grid for the west panel
    westPanel.setBackground(new Color(120,180,190)); //set the color to the west panel
}
```

- III. The `actionPerformed(ActionEvent event)` function is responsible for interpreting and executing each action initiated from the Graphical User Interface (GUI) whenever a button is pressed.

```
//Creating action listener method to be able to interact with the program from GUI
public void actionPerformed(ActionEvent event)
{
    String command = event.getActionCommand(); //making action listener to read string commands
    if (command.equals("Add New Phone")) // add if statement to check what button was pressed
    {
        captureInfo("Phone"); //calling the "captureInfo" method which has an parameter
    }
    if (command.equals("Add New MP3"))
    {
        captureInfo("Mp3");
    }
    if (command.equals("Clear")) // add if statement to check what button was pressed
    {
        Clear(); //calling the "Clear" method which dosen't have a parameter
    }
    if (command.equals("Display All")) // add if statement to check if "Display All" button was p
    {
        System.out.println(numberOfGadgets() + " Gadgets in library"); // print statement which
        //shows how many gadgets are in arrayList
        System.out.println(); // empty print statement
        System.out.println("The list of gageets is : " + gadgets ); // print statement with list of
        System.out.println("-----");
        System.out.println();
        for (Gadget g : gadgets) //Looping through the gadget list
        {
            Gadget currentGad = g; // assigning the current gadget to a variable
            if (currentGad instanceof Mobile) // checking if current gadget belongs to Mobile class
            {
                Mobile mobile = (Mobile) currentGad; // casting the current gadget to Mobile
                //so we can acces the mobile methods from the gui class
                mobile.displayMobAtributes(); // calling displayMobAtributes method
                //for the mobile item in arryList through casing from mobile class to display mobil
            }
            if (currentGad instanceof Mp3) // checking if current gadget belongs to Mp3 class
            {
                Mp3 mp3 = (Mp3) currentGad; // casting the current gadget to Mp3
                mp3.displayMp3Atributes(); // calling displayMp3Atributes method
            }
        }
    }
    if (command.equals("Call")) // add if statement to check if "Call" button was pressed
    {
        checkDuration(); // calling the check duration method
    }
    if (command.equals("Download"))
    {

```

- IV. The `main(String[] args)` method serves as the entry point to the program, allowing access to it directly from the command prompt of the machine.

```
public static void main(String[] args) // creating main class to open the app from cmd
{
    EShopGUI EShopGUI = new EShopGUI(); //creating a new object EShopGUI of type EShopGUI
}
```

## b) Methods in the Gadget class

### I. Constructor method of Gadget class.

```
public Gadget(String aModel, double aPrice, int aWeight, String aSize)
//This is a constructor method for the Gadget class
{
    // initialise instance variables
    model = aModel;
    price = aPrice;
    weight = aWeight;
    size = aSize;
}
```

### II. The showModel() method is involved in the main methods such as: phone call and download music.

```
public Gadget(String aModel, double aPrice, int aWeight, String aSize)
//This is a constructor method for the Gadget class
{
    // initialise instance variables
    model = aModel;
    price = aPrice;
    weight = aWeight;
    size = aSize;
}
```

### III. The printGadget function is a part of the displayAll method.

```
//Print the gadget attributes list
public void printGadget()
{
    System.out.println("Model: " + model);
    System.out.println("Price: " + price + " £");
    System.out.println("Weight: " + weight + " grams");
    System.out.println("Size: " + size);
}
```



### c) Methods in the Mobile class

- I. Constructor method of Mobile class that inherits Gadget constructor method.

```
public Mobile(String aModel, double aPrice, int aWeight,
String aSize, float aCredit)
{
    /* initialise instance variables for mobile class and
    adding the variables from the super class gadget*/
    super(aModel, aPrice, aWeight, aSize );
    credit = aCredit;
}
```

- II. The displayMobAttributes function is a component of the displayAll method, accessed through casting to show the initial memory of the phone.

```
public void displayMobAttributes()
{
    System.out.println("The Mobile Characteristics are:");
    super.printGadget();
    System.out.println("Your balance is: " + credit + " min");
    System.out.println("");
}
```

- III. The getGredit method can be seen inside the Call method to get the value of the credit.

```
public float getCredit()
{
    return credit;
}
```

- IV. The setGredit method can be seen inside the Call method to set a new value after that call is done.

```
public void setCredit(float aCredit)
{
    credit = aCredit;
}
```

#### d) Methods in the Mp3 class

- I. Constructor method of Mp3 class that inherits Gadget constructor method.

```
public Mp3(String aModel, double aPrice, int aWeight,
String aSize, float aMemory)
{
    /* initialise instance variables for Mp3 class and
    adding the variables from the super class gadget*/
    super(aModel, aPrice, aWeight, aSize );
    memory = aMemory;
}
```

- II. The displayMp3Attributes function is a component of the displayAll method, accessed through casting to show the initial memory of the Mp3.

```
//Printing the attributes for Mp3
public void displayMp3Attributes()
{
    System.out.println("The Mp3 Characteristics are:");
    super.printGadget();
    System.out.println("The available space is: " + memory + " GB");
    System.out.println();
}
```

- III. The getMemory method is utilized within the Download method to retrieve the memory value.

```
public float getMemory()
{
    return memory;
}
```

- IV. The setMemory method is utilized within the Download method to rewrite the memory value after downloading music.

```
public void setMemory(float aMemory)
{
    memory = aMemory;
}
```

## 6. The evidence of the following testing.

### Test 1: Adding a mobile to the Array List

The image shows two windows from a Java IDE. The top window is titled 'E-Shop' and contains a GUI for a gadget store. It has a menu bar with 'File' and 'Help'. The main area is divided into a left sidebar with 'Store' and 'Library' sections, and a main content area. The 'Store' section lists attributes: Model (Galaxy s23), Price (452.00), Size (108 x 84 x 10), Weight (173), Credit (15.62), Memory, Phone No., Duration, Download, and DisplayNumber. The 'Library' section is currently empty. The main content area has buttons: 'Add New Phone', 'Add New MP3', 'Clear', 'Display All', 'Call', and 'Download'. The bottom window is titled 'BlueJ: Terminal Window - Gadget2' and shows the output of the 'Display All' button click. It displays '0 Gadgets in library' and 'The list of gadgets is : []'. A status bar at the bottom of the terminal window says 'Can only enter input while your program is running'.

Welcome to Gadget Shop	
Model:	Galaxy s23
Price:	452.00
Size:	108 x 84 x 10
Weight	173
Credit	15.62
Memory	
Phone No.	
Duration	
Download	
DisplayNumber	

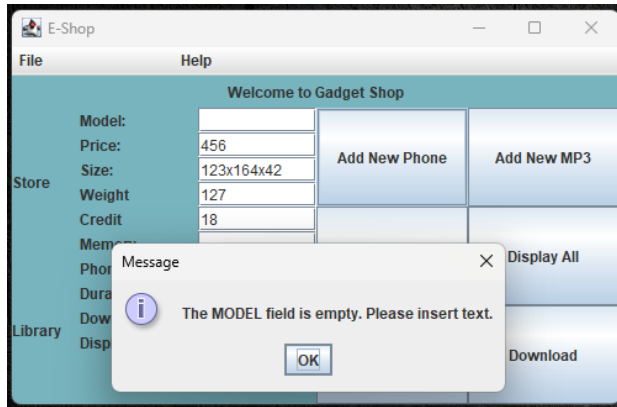
Buttons: Add New Phone, Add New MP3, Clear, Display All, Call, Download

Terminal Output:

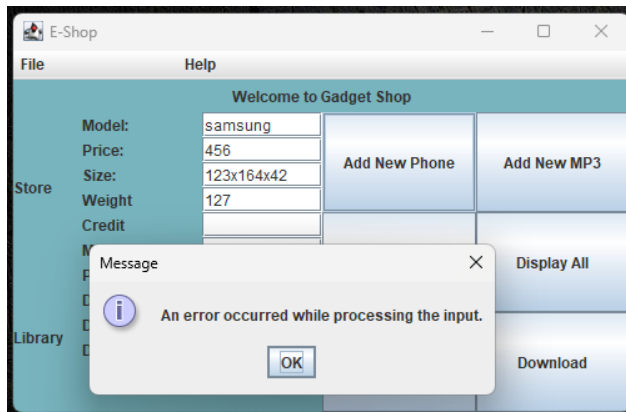
```
Options
0 Gadgets in library
The list of gadgets is : []
```

Can only enter input while your program is running

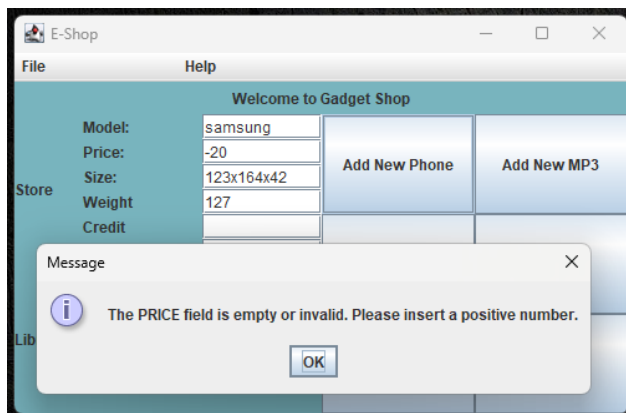
Above is represented the GUI and command prompt when it was turned on and pressed the Display all button so you can see that the gadgets library is empty.



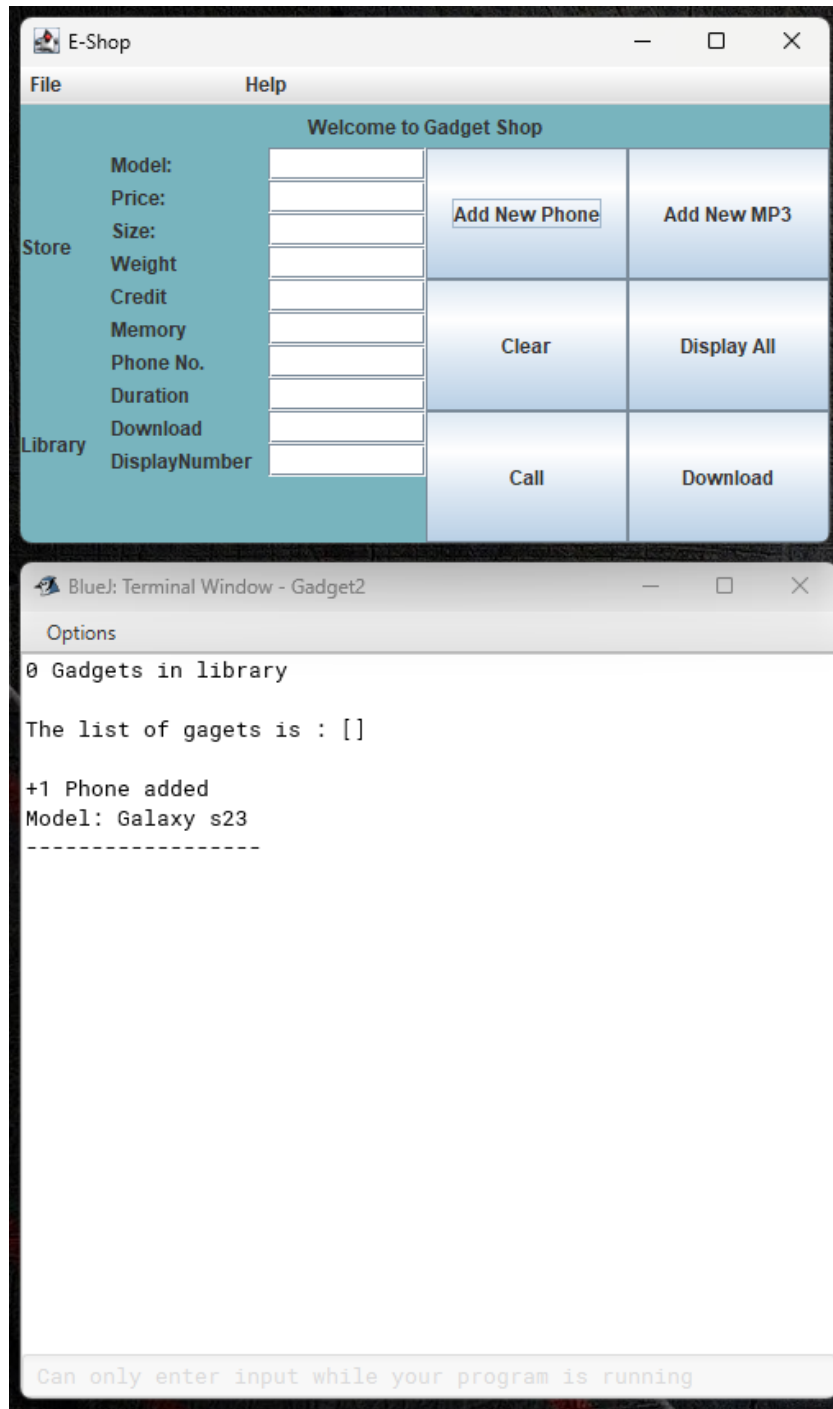
Here, a pop-up message appears indicating that the model field is empty, prompting the user to insert text.



The credit field is empty so we have to fill the empty space.



The price is less than zero so we have to insert a positive value



This picture shows the GUI and the command prompt after we push the button add Mobile so you can see that a mobile is added successfully to the list and the text field in the GUI is empty now.

## Test 2: Adding an Mp3 to the ArrayList

The image shows two windows from a Java IDE. The top window is titled "E-Shop" and contains a form for adding new devices. The form has a menu bar with "File" and "Help". Below the menu bar is a header "Welcome to Gadget Shop". The form is divided into two main sections: "Store" and "Library". The "Store" section contains fields for "Model:", "Price:", "Size:", "Weight", "Credit", "Memory", "Phone No.", and "Duration". The "Library" section contains fields for "Download" and "DisplayNumber". To the right of these fields are four buttons: "Add New Phone", "Add New MP3", "Clear", and "Display All". Below these buttons are two more buttons: "Call" and "Download". The bottom window is titled "BlueJ: Terminal Window - Gadget2" and shows the output of the program. It displays "0 Gadgets in library" and "The list of gadgets is : []". At the bottom of the terminal, there is a message: "Can only enter input while your program is running".

Welcome to Gadget Shop	
Model:	Sony f3
Price:	362.00
Size:	108 x 84 x 10
Weight	173
Credit	
Memory	128
Phone No.	
Duration	
Download	
DisplayNumber	

Buttons: Add New Phone, Add New MP3, Clear, Display All, Call, Download

BlueJ: Terminal Window - Gadget2

Options

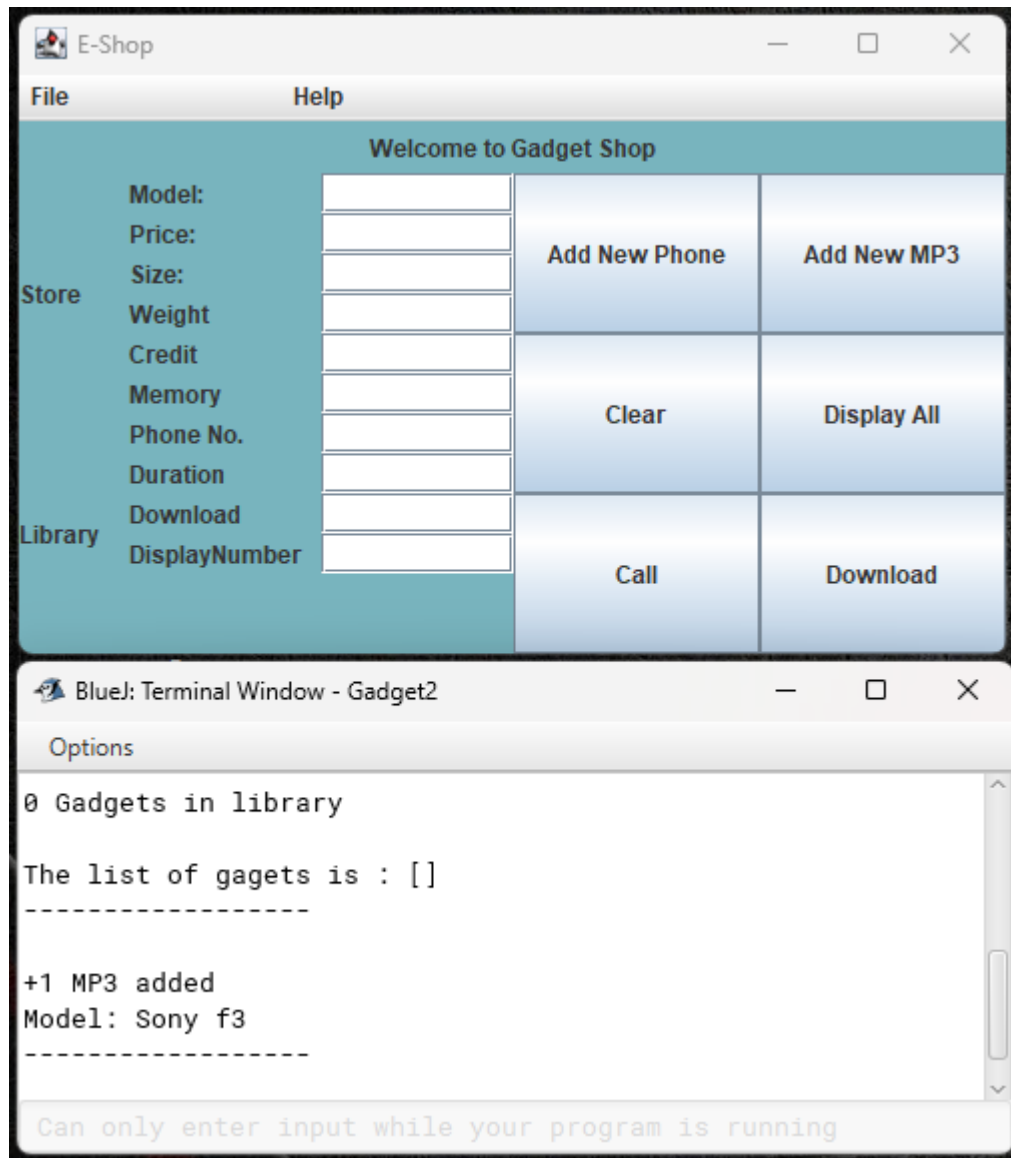
0 Gadgets in library

The list of gadgets is : []

-----

Can only enter input while your program is running

View the form with all text fields correctly filled before adding any device to the ArrayList. If incorrect information is entered when adding a device, a message will appear on the screen, similar to what we observed when adding a mobile.



The view displaying the successful addition of an Mp3 device.

Test 3: Display all the information.

```
BlueJ: Terminal Window - Gadget2
Options
+1 Phone added
Model: Galaxy s23
-----

+1 MP3 added
Model: Galaxy s23
-----

2 Gadgets in library

The list of gageets is : [Mobile@91d8ebe, Mp3@313025f1]
-----

The Mobile Characteristics are:
Model: Galaxy s23
Price: 452.0 £
Weight: 173 grams
Size: 108 x 84 x 10
Your balance is: 15.62 min

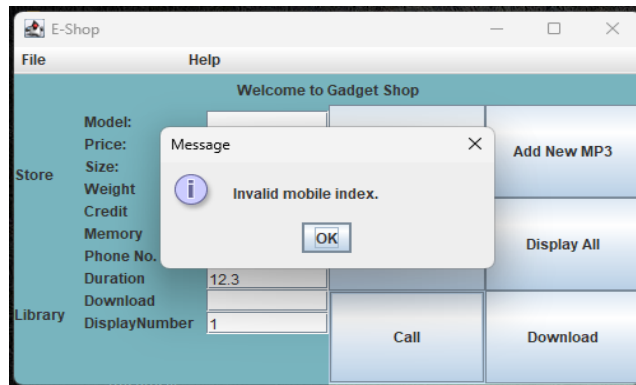
The Mp3 Characteristics are:
Model: Galaxy s23
Price: 162.5 £
Weight: 123 grams
Size: 88 x 64 x 8
The available space is: 128.0 GB

Can only enter input while your program is running
```

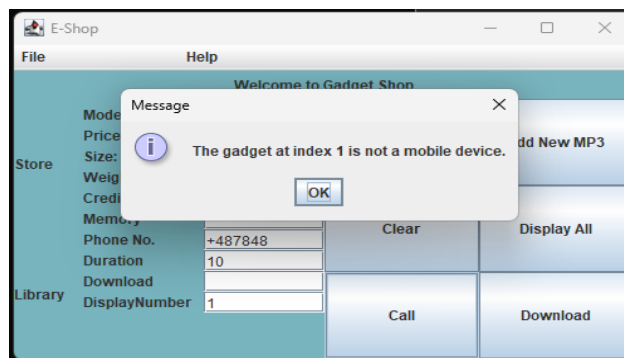
You can observe that the “Display All” function is operational as it successfully printed two devices from the list along with their respective characteristics.



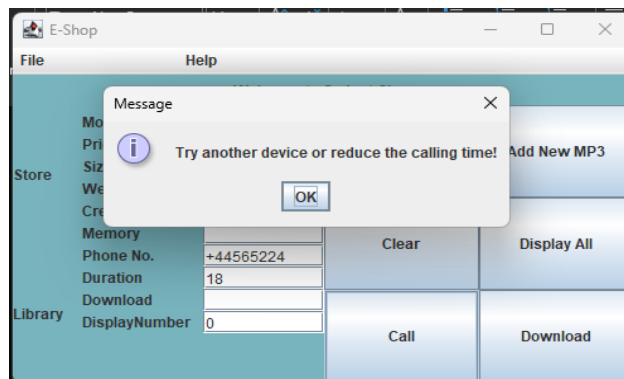
## Test 4: Make a Call



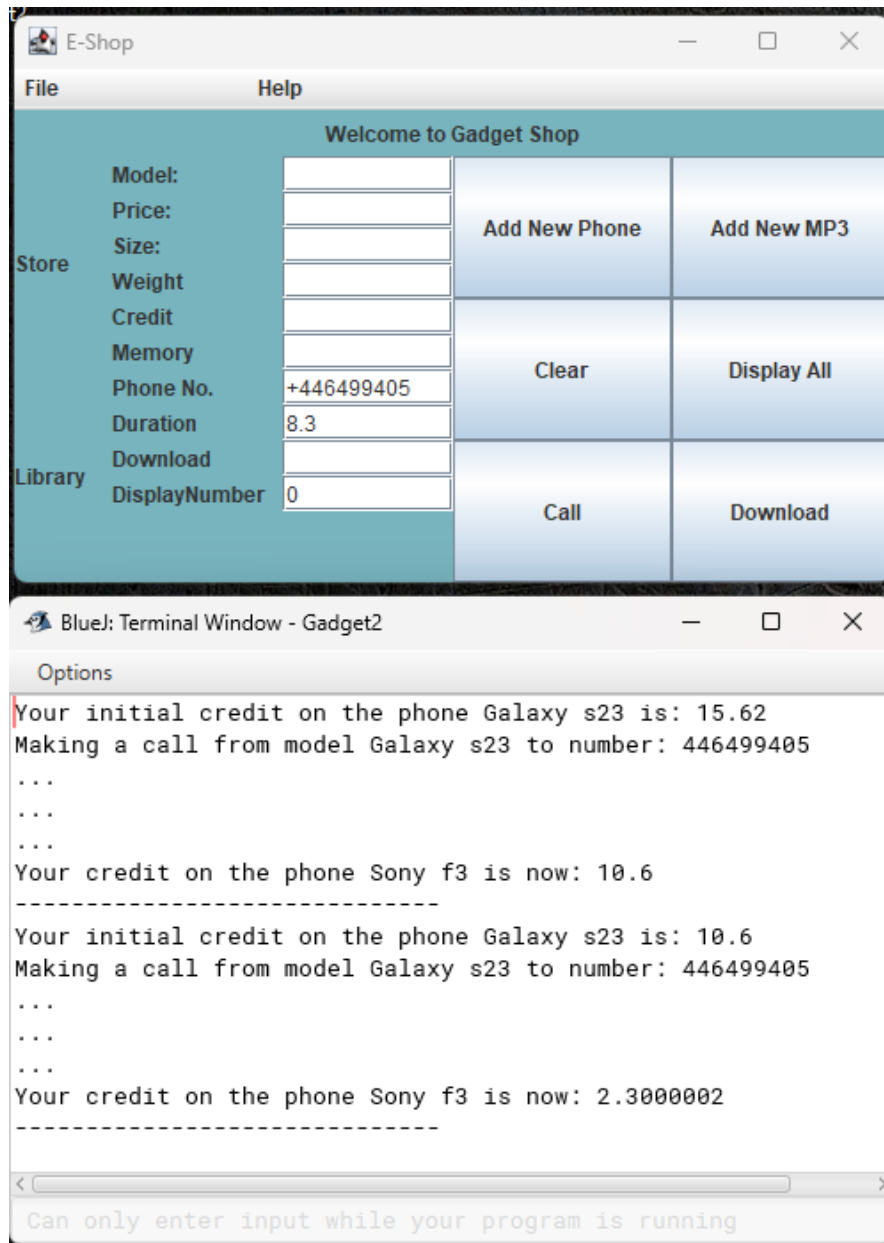
This scenario occurs when there is only one mobile device in the library, and we have selected a device number that exceeds the total number of devices currently stored in the list.



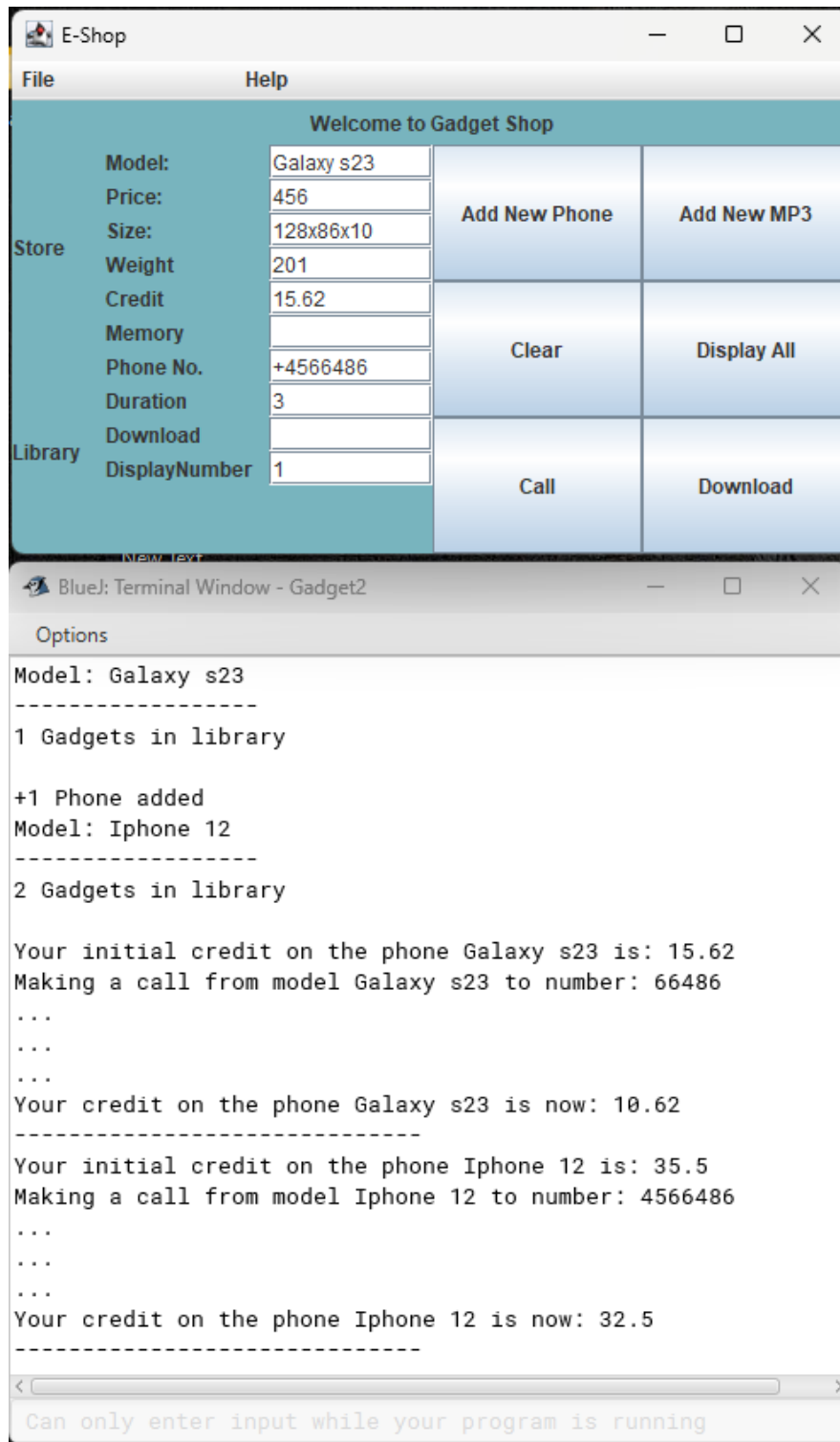
This is the scenario when the currently selected device is not a mobile phone.



This is the case when the credit is less than the duration of the call.



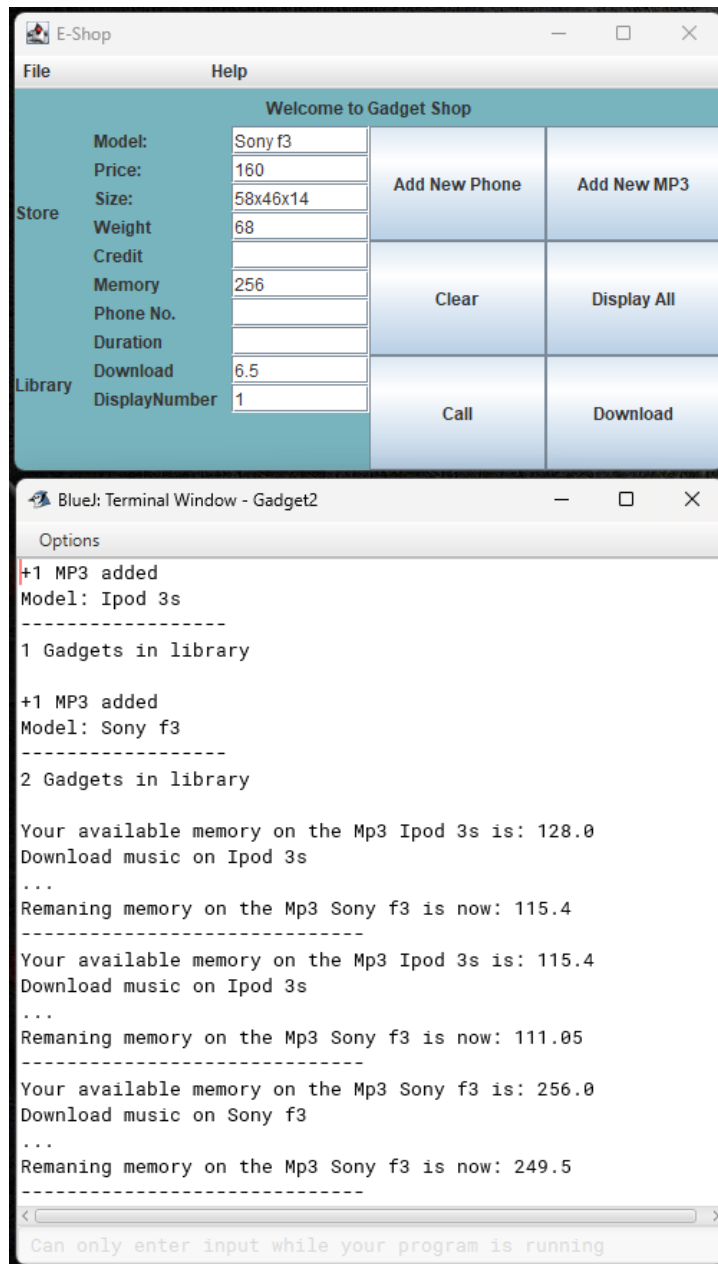
In this scenario, with only one mobile device in the list and multiple calls are being made. You can observe that the credit is decreasing with each call, and a new value is assigned to it every time a call is made.



A case where there are multiple mobile devices, and each call updates the information automatically, ensuring that the data is always current and accurate.

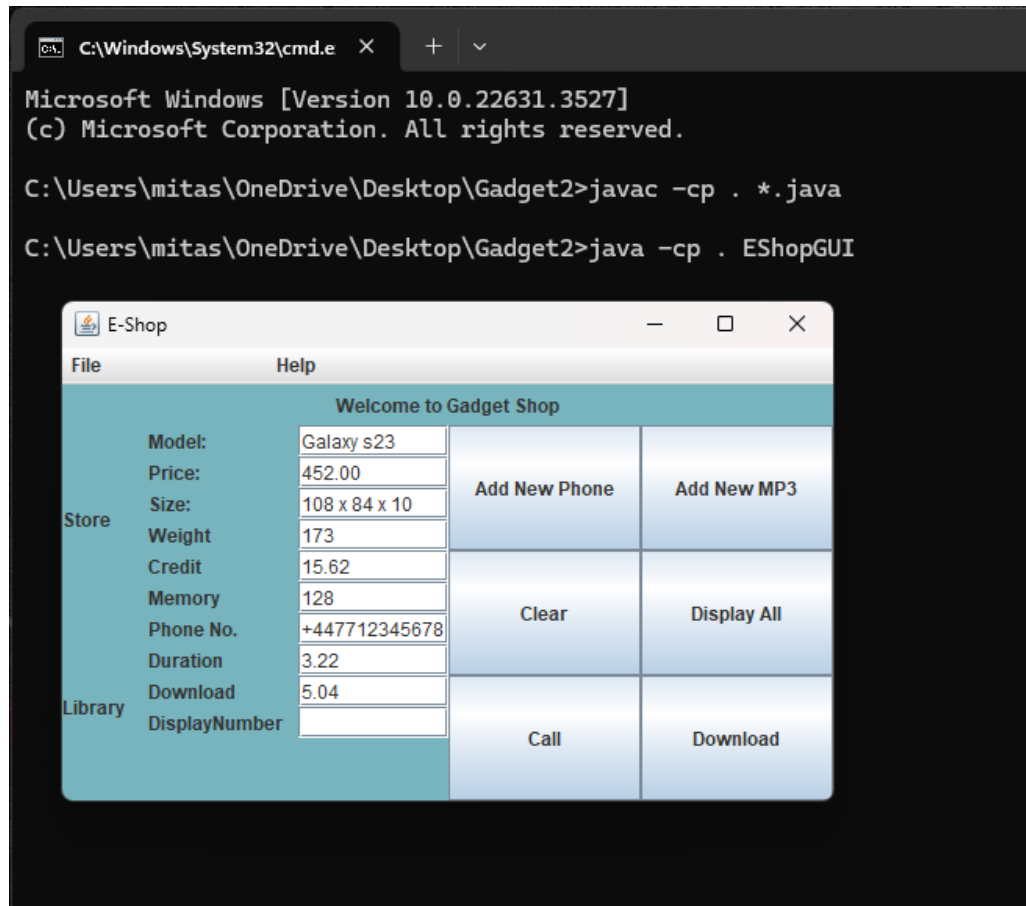
## Test 5: Download music.

Similar to the messages generated when an error occurs during the “make call” function on mobile devices when attempting to download music on MP3 devices, these messages appear to assist the user in understanding any mistakes made.



In a test with two Mp3s, attempting to download music reveals that the information about the memory is automatically updated for each device with every download operation.

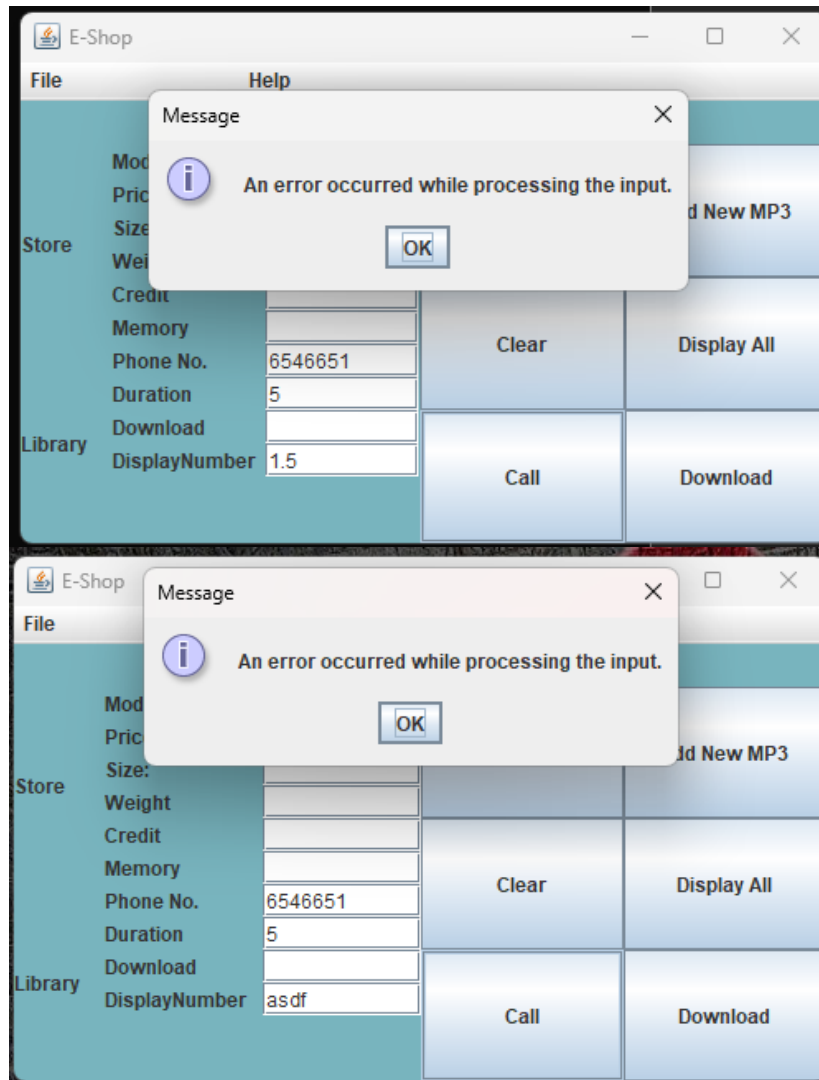
Test 6: Open the program from the Command Prompt.



I encountered some issues when attempting to open my program from the command prompt, which I'll illustrate in the following section with appropriate images. Finally after some research and with some help from the tutor it works.

Test 7: Show that appropriate dialog boxes appear when unsuitable values are entered for the display number.

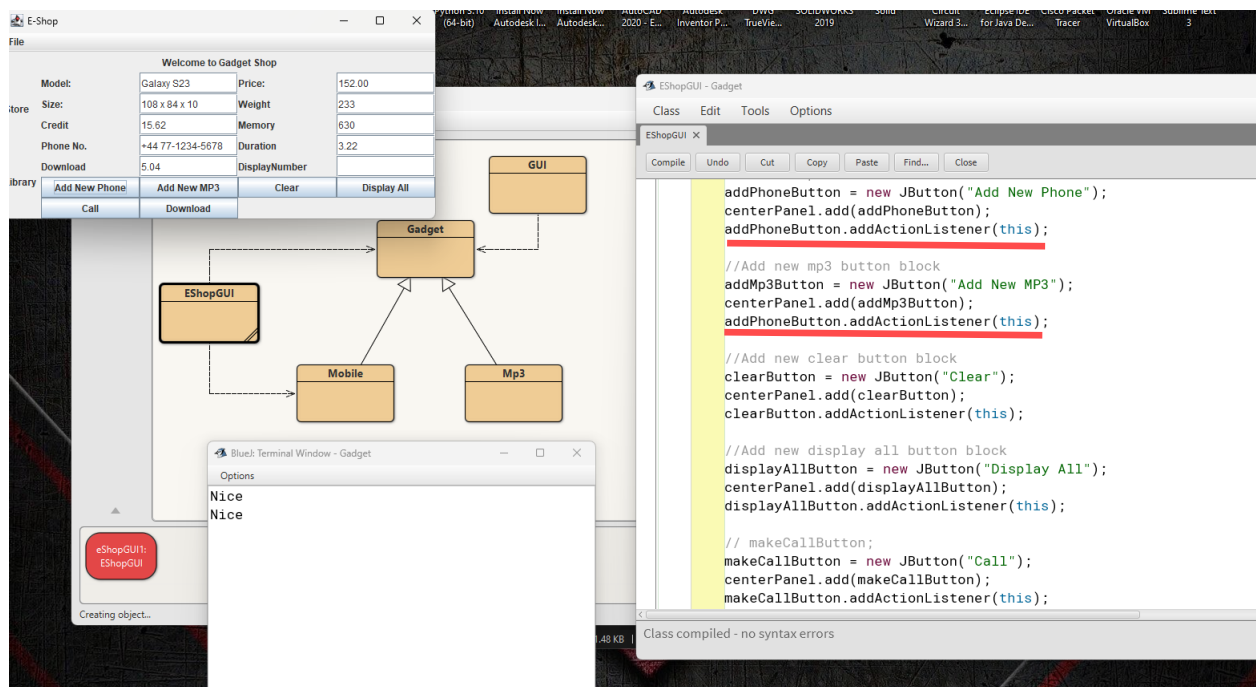
In all previous tests, I've demonstrated that when a new device is added and any text field is left empty, or if incorrect data is entered, such as a string in the price field or a negative number, a frame with a helpful message appears on the screen. Similarly, if the display number field contains a string, float, or integer less than 0, or exceeds the total number of gadgets in the list, the same frame with a helpful message appears.



## 7. Errors

While working on this project, I encountered numerous issues and bugs with the program. However, I've come to realize that these challenges are an inherent aspect of learning. Each problem serves as a unique challenge, pushing me to find the correct way to address it.

- a) At the outset of the project, I encountered an issue where adding a new phone resulted in duplication every time the button was pressed. The problem stemmed from my attempt to save time by copy-pasting the same function twice. However, I overlooked updating all the parameters within that block for the MP3



```
//Add new phone button block
addPhoneButton = new JButton("Add New Phone"); //create new button
eastPanel.add(addPhoneButton); //add button to the east panel
addPhoneButton.addActionListener(this); //connect button to action

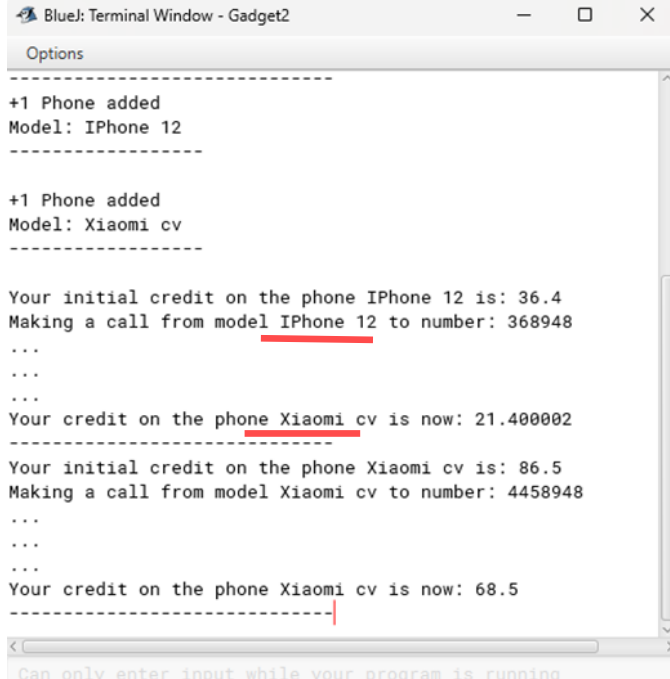
//Add new mp3 button block
addMp3Button = new JButton("Add New MP3");
eastPanel.add(addMp3Button);
addMp3Button.addActionListener(this);

//Add new clear button block
clearButton = new JButton("Clear");
eastPanel.add(clearButton);
clearButton.addActionListener(this);
```

Here the problem was solved.

- b) Another issue arose due to an error in the code. Each time a call from the phone was made, the command prompt displayed the last model added to the list inadvertently.

```
if (duration >= 0 && duration <= currCredit)
{
    System.out.println("Your credit on " + model + "is now: " + credit);
    currCredit -= duration;
    mobile.setCredit(currCredit);
    System.out.println("Making a call from model " + mobile.showModel() + " to number: " + phoneNumber );
    System.out.println("...");
    System.out.println("Your credit on " + model + "is now: " + credit);
}
else
{
    JOptionPane.showMessageDialog(frame, "Try another device or reduce the calling time!");
}
```



```
+1 Phone added
Model: iPhone 12
-----

+1 Phone added
Model: Xiaomi cv
-----

Your initial credit on the phone iPhone 12 is: 36.4
Making a call from model iPhone 12 to number: 368948
...
...
Your credit on the phone Xiaomi cv is now: 21.400002
-----

Your initial credit on the phone Xiaomi cv is: 86.5
Making a call from model Xiaomi cv to number: 4458948
...
...
Your credit on the phone Xiaomi cv is now: 68.5
-----

Can only enter input while your program is running
```

The following pictures demonstrate that the issue was resolved in the code, as evidenced by the information printed in the BlueJ terminal.

```
if (duration >= 0 && duration <= currCredit) // checking if the duration is less than the existing credit of th
{
    System.out.println("Your initial credit on the phone " + mobile.showModel() + " is: " + currCredit); // prin
    currCredit -= duration; //subtract duration from initial credit
    mobile.setCredit(currCredit); // assign new value after the subtraction
    System.out.println("Making a call from model " + mobile.showModel() + " to number: " + phoneNumber ); // pri
    System.out.println("...");
    System.out.println("...");
    System.out.println("...");
    System.out.println("Your credit on the phone " + mobile.showModel() + " is now: " + currCredit ); // display
    System.out.println("-----");
}
```



```
BlueJ: Terminal Window - Gadget2
Options
Model: Galaxy s23
-----
1 Gadgets in library

+1 Phone added
Model: Iphone 12
-----
2 Gadgets in library

Your initial credit on the phone Galaxy s23 is: 15.62
Making a call from model Galaxy s23 to number: 66486
...
...
...
Your credit on the phone Galaxy s23 is now: 10.62
-----
Your initial credit on the phone Iphone 12 is: 35.5
Making a call from model Iphone 12 to number: 4566486
...
...
...
Your credit on the phone Iphone 12 is now: 32.5
-----
Can only enter input while your program is running
```

- c) This was another issue encountered while attempting to open the program through the terminal. My machine indicated that it couldn't find the main class to launch the program. However, as depicted, the main class is located within the EShopGUI class and is correctly written.

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mitas\OneDrive\Desktop\Gadget2>javac *.java

C:\Users\mitas\OneDrive\Desktop\Gadget2>java Main.java
Error: Could not find or load main class Main.java
Caused by: java.lang.ClassNotFoundException: Main.java

C:\Users\mitas\OneDrive\Desktop\Gadget2>
```

```
public static void main(String[] args) // creating main class to open the app from cmd
{
    EShopGUI EShopGUI = new EShopGUI(); //creating a new object EShopGUI of type EShopGUI
}
```

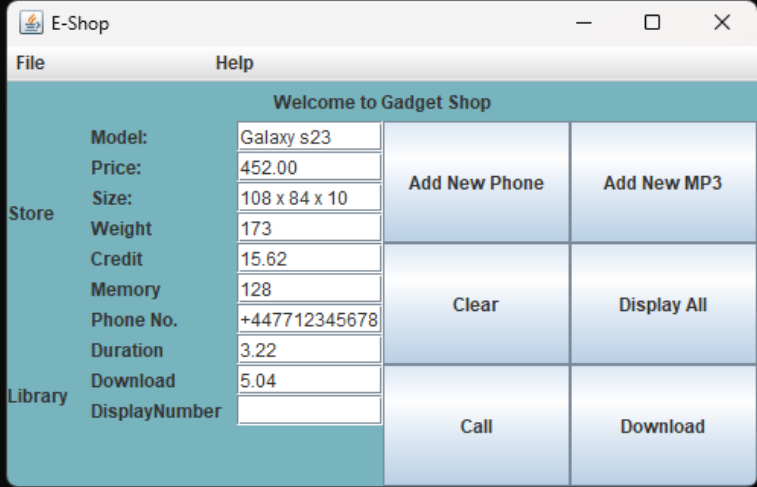
After some research and making several unsuccessful attempts, along with receiving assistance from a tutor, I finally managed to open it through the terminal, because in the Windows operating system you have to follow some different steps and bellow you can see first and last step.

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

C:\Users\mitas\OneDrive\Desktop\Gadget2>javac *.java

C:\Users\mitas\OneDrive\Desktop\Gadget2>java Main.java
Error: Could not find or load main class Main.java
Caused by: java.lang.ClassNotFoundException: Main.java

C:\Users\mitas\OneDrive\Desktop\Gadget2>java -cp . EShopGUI
```



Welcome to Gadget Shop			
Store	Model:	Galaxy s23	
	Price:	452.00	
	Size:	108 x 84 x 10	
	Weight	173	
	Credit	15.62	
	Memory	128	
	Phone No.	+447712345678	
	Duration	3.22	
	Download	5.04	
Library	DisplayNumber		
		Add New Phone	Add New MP3
		Clear	Display All
		Call	Download

These are just a few examples of the challenges I faced while working on this program. However, overcoming them has bolstered my confidence, and I now feel better prepared to tackle similar obstacles in future projects.

## 8. Conclusion

This project was a rewarding challenge and provided a valuable learning experience. I thoroughly enjoyed working on it and gained new skills, particularly in GUI creation. Even it seems simple, GUIs require considerable effort. It's essential to think from the perspective of a user to ensure the interface is user-friendly and intuitive. It also demands a solid understanding of programming logic to ensure the methods function correctly.

In the process of creating this GUI, I familiarized myself with various tactics and methods, such as the 'try' method, which enables the capturing of errors from users without crashing the program.

Another crucial concept I learned was inheritance, which facilitates subclasses extending from superclasses and utilizing their methods. This concept I implemented in my project.

Additionally, I employed method overriding and overloading, enabling access to different functions from the Gadget superclass.

However, the most impactful concept for me was casting, which allowed me to convert items from the Gadget type to Mobile or Mp3 types and manipulate them. In hindsight, I believe this concept should have been studied earlier, as it significantly contributed to my project development and could have potentially mitigated delays.

## 9. Appendix

### I. EShopGUI CLASS

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.ArrayList;

public class EShopGUI implements ActionListener
{
    private JFrame frame; //Variable to create the frame

    // Variables for the text field in type JTextField
    private JTextField theModel;
    private JTextField thePrice;
    private JTextField theWeight;
    private JTextField theSize;
    private JTextField theCredit;
    private JTextField theMemory;
    private JTextField thePhoneNumb;
    private JTextField theDuration;
    private JTextField Download;
    private JTextField DisplayNumber;

    //Variables for buttons in type JButton
    private JButton addPhoneButton;
    private JButton addMp3Button;
    private JButton clearButton;
    private JButton displayAllButton;
    private JButton makeCallButton;
    private JButton downloadMusicButton;

    //Creating global variables to be stored in our arrayList
    private String model;
    private double price;
    private String size;
    private int weight;
```

```

private float memory;
private float credit;
private int phoneNumber;
private float duration;
private float download;
private int displayNumber = -1;

//Creating a variable of arraylist type of string
private ArrayList<Gadget> gadgets;

public EShopGUI()
{
    gadgets = new ArrayList<Gadget>(); // creating the gadget array list
    drawGUI(); // calling the draw GUI to set the window for the GUI
}

public void drawGUI()
{
    frame = new JFrame("E-Shop"); //create the frame window
    frame.setLocation(400,100); //set the location of the frame on the screen

    JMenuBar menuBar = new JMenuBar(); //Create new empty menu bar
    frame.setJMenuBar(menuBar); //set the menu bar to the frame

    //File block
    JMenu fileMenu = new JMenu("File"); // create the file menu
    menuBar.add(fileMenu); // add the file menu to the menu bar

    JMenuItem newFileItem = new JMenuItem("New File"); //create a new item
    newFileItem.addActionListener(this); //add the "new file" item to action listener to
provide the comand
    fileMenu.add(newFileItem); // add the "new file" item to the file in menu bar menu

    JMenuItem openFileItem = new JMenuItem("Open");
    openFileItem.addActionListener(this);
    fileMenu.add(openFileItem);

    menuBar.add(Box.createHorizontalStrut(100)); // add a space between the 2 items in
menu bar

```

```
//menuBar.add(Box.createHorizontalGlue()); // add maximum available space between 2
menu items
```

```
//Help block
```

```
JMenu helpMenu = new JMenu("Help"); // create the file menu
menuBar.add(helpMenu); // add the file menu to the menu bar
```

```
Container contentPane = frame.getContentPane();//create the spase to be filed with
butons and fields
```

```
BorderLayout BorderLayout = new BorderLayout(); //create new border layout
contentPane.setLayout(BorderLayout); //set the border layout for the content panel
```

```
//Panel for the North block
```

```
JPanel northPanel = new JPanel(); // crete a new panel
contentPane.add(northPanel, BorderLayout.NORTH); // add the north border to the top
of the window panel
```

```
northPanel.add(new JLabel("Welcome to Gadget Shop")); // add a label to north borer
northPanel.setBackground(new Color(120,180,190)); //set the color to the north panel
```

```
//Panel for the West block
```

```
JPanel westPanel = new JPanel(); //create a new panel in the left side of the window
contentPane.add(westPanel, BorderLayout.WEST); // add the west panel to the border
layout
```

```
GridLayout westGridLayout = new GridLayout(2,1); //create grid layout in the west
bordr of the layout
```

```
westPanel.setLayout(westGridLayout); //seting the grid for the west panel
westPanel.setBackground(new Color(120,180,190)); //set the color to the west panel
westPanel.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 15)); //set a space of 15
pixels between west and center panel
```

```
westPanel.add(new JLabel("Store")); // set a label in the west panel
westPanel.add(new JLabel("Library")); // set a label in the west panel
```

```
//Panel for the Center block
```

```
JPanel centerPanel = new JPanel(); //create a new panel in the center of the window
contentPane.add(centerPanel, BorderLayout.CENTER); //add border layout to the center
panel
```

```
centerPanel.setBackground(new Color(120,180,190)); //set color to center panel
```

```
GridLayout gridCenterLayout = new GridLayout(12,2); //create grid layout in the west  
border of the layout
```

```
centerPanel.setLayout(gridCenterLayout); //set the grid layout to center panel
```

```
//The "model" field block which contains the label and the text field
```

```
centerPanel.add(new JLabel("Model:")); //add the "model" label to center panel
```

```
//The "model" text field block
```

```
theModel = new JTextField("Galaxy s23"); //create new text box
```

```
centerPanel.add(theModel); //add the text box to the center panel
```

```
//The "price" field block which contains the label and the field
```

```
centerPanel.add(new JLabel("Price:")); //add the "price" label to center panel
```

```
//The "price" text field block
```

```
thePrice = new JTextField("452.00");
```

```
centerPanel.add(thePrice);
```

```
//The "size" field block which contains the label and the field
```

```
centerPanel.add(new JLabel("Size:")); //add the "size" label to center panel
```

```
//The "size" text field block
```

```
theSize = new JTextField("108 x 84 x 10");
```

```
centerPanel.add(theSize);
```

```
//The "weight" field block which contains the label and the field
```

```
centerPanel.add(new JLabel("Weight:"));
```

```
//The "weight" text field block
```

```
theWeight = new JTextField("173");
```

```
centerPanel.add(theWeight);
```

```
//The "credit" field block which contains the label and the field
```

```
centerPanel.add(new JLabel("Credit:"));
```

```
//The "credit" text field block
```

```
theCredit = new JTextField("15.62");
```

```
centerPanel.add(theCredit);
```

```
//The "memory" field block which contains the label and the field
```

```
centerPanel.add(new JLabel("Memory:"));
```

```
//The "memory" text field block
```

```
theMemory = new JTextField("128");
```

```
centerPanel.add(theMemory);
```

```

//The "phone number" field block which contains the label and the field
centerPanel.add(new JLabel("Phone No.));
//The "phone number" text field block
thePhoneNumb = new JTextField("+447712345678");
centerPanel.add(thePhoneNumb);

//The "duration of the song" field block which contains the label and the field
centerPanel.add(new JLabel("Duration"));
//The "duration" text field block
theDuration = new JTextField("3.22");
centerPanel.add(theDuration);

//The "download" field block which contains the label and the field
centerPanel.add(new JLabel("Download"));
//The "download" text field block
Download = new JTextField("5.04");
centerPanel.add(Download);

//The "display number" field block which contains the label and the field
centerPanel.add(new JLabel("DisplayNumber"));
//The "display number" text field block
DisplayNumber = new JTextField();
centerPanel.add(DisplayNumber);

JPanel eastPanel = new JPanel(); //create a new panel in the left side of the window
contentPane.add(eastPanel, BorderLayout.EAST); // add the west panel to the border
layout
GridLayout eastGridLayout = new GridLayout(3,2); //create grid layout in the west
bordr of the layout
eastPanel.setLayout(eastGridLayout); //seting the grid for the west panel
eastPanel.setBackground(new Color(210,249,241));

//Add new phone button block
addPhoneButton = new JButton("Add New Phone"); //create new button called "add
new phone"
eastPanel.add(addPhoneButton); //add button to the east panel
addPhoneButton.addActionListener(this); //connect button to action listener

//Add new mp3 button block
addMp3Button = new JButton("Add New MP3");

```



```

eastPanel.add(addMp3Button);
addMp3Button.addActionListener(this);

//Add new clear button block
clearButton = new JButton("Clear");
eastPanel.add(clearButton);
clearButton.addActionListener(this);

//Add new display all button block
displayAllButton = new JButton("Display All");
eastPanel.add(displayAllButton);
displayAllButton.addActionListener(this);

// makeCallButton;
makeCallButton = new JButton("Call");
eastPanel.add(makeCallButton);
makeCallButton.addActionListener(this);

// downloadMusicButton;
downloadMusicButton = new JButton("Download");
eastPanel.add(downloadMusicButton);
downloadMusicButton.addActionListener(this);

frame.pack(); //set minimized frame
frame.setVisible(true); // set the visibility of the frame when start the program
}

//Creating action listener method to be able to interact with the program from GUI
public void actionPerformed(ActionEvent event)
{
    String command = event.getActionCommand(); //making action listener to read string
    commands
    if (command.equals("Add New Phone")) // add if statement to check what button was
    pressed
    {
        captureInfo("Phone"); //calling the "captureInfo" method which has an parameter
    }
    if (command.equals("Add New MP3"))
    {
        captureInfo("Mp3");
    }
}

```

```

        if (command.equals("Clear")) // add if statement to check what button was pressed
        {
            Clear(); //calling the "Clear" method which dosen't have a parameter
        }
        if (command.equals("Display All")) // add if statement to check if "Display All" button
was pressed
        {
            System.out.println(numberOfGadgets() + " Gadgets in library"); // print statement
which
            //shows how many gadgets are in arrayList
            System.out.println(); // empty print statement
            System.out.println("The list of gageets is : " + gadgets ); // print statement with list of
the gadgets
            System.out.println("-----");
            System.out.println();
            for (Gadget g : gadgets) //Looping through the gadget list
            {
                Gadget currentGad = g; // assigning the current gadget to a variable
                if (currentGad instanceof Mobile) // cheching if current gadget belongs to Mobile
class
                {
                    Mobile mobile = (Mobile) currentGad; // casting the current gadget to Mobile
                    //so we can acces the mobile methods from the gui class
                    mobile.displayMobAtributes(); // calling displayMobAtributes method
                    //for the mobile item in arryList through casing from mobile class to display
mobiles atributes
                }
                if (currentGad instanceof Mp3) // cheching if current gadget belongs to Mp3 class
                {
                    Mp3 mp3 = (Mp3) currentGad; // casting the current gadget to Mp3
                    mp3.displayMp3Atributes(); // calling displayMp3Atributes method
                }
            }
        }
        if (command.equals("Call")) // add if statement to check if "Call" button was pressed
        {
            checkDuration(); // calling the check duration method
        }
        if (command.equals("Download"))
        {
            checkMemory();

```

```

    }
}

public int numberOfGadgets() // number of gadgets method
{
    return gadgets.size(); //returning the size of the arrayList by using the
    //built-in method on lists .size()
}

private void captureInfo(String gadgetType) // capture method which checks if every text
field is completed correctly
//and it is using a parameter to check which gadget to add to gadgetList
{
    try //using try statement to make the program more error stable and user friendly
    {
        model = theModel.getText(); //read the text field from GUI and assigning it to a
variable
        if (model.isEmpty()) //Check if the text field is empty
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The MODEL field is empty. Please
insert text.");
            return; //break the program and return to the invoked method
        }

        price = Double.parseDouble(thePrice.getText()); //read the text field from GUI,
//convert it from string to double and assigning it to variable price
        if (price <= 0) //check if the price is less than 0
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The PRICE field is empty or invalid.
Please insert a positive number.");
            return; //break the program and return to the invoked method
        }

        size = theSize.getText(); //read the text field from GUI and assigning it to variable size
        if (size.isEmpty()) //Check if the text field is empty
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The SIZE field is empty. Please insert
text.");

```

```

        return; //break the program and return to the invoked method
    }

    weight = Integer.parseInt(theWeight.getText()); //read the text field from GUI,
    //convert it from string to integer type and assigning it to variable weight
    if (weight <= 0) //check if the price is less than 0
    {
        //pop-up an informative message for user
        JOptionPane.showMessageDialog(frame, "The WEIGHT field is empty or
invalid. Please insert a positive number.");
        return; //break the program and return to the invoked method
    }

    if (gadgetType.equals("Phone")) //checking gadget type if it is a phone
    {
        //add a new attribute if the gadget it's a phone
        credit = Float.parseFloat(theCredit.getText()); //read the text field from GUI,
        //convert it from string to float type and assigning it to variable credit
        if (credit <= 0) //check if the credit is less than 0
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The CREDIT field is empty or
invalid. Please insert a positive number.");
            return; //break the program and return to the invoked method
        }
        addPhone(); //using addPhone method to add new phone to gadget list
        Clear(); //using clear method to clear the fields
    }
    else if (gadgetType.equals("Mp3")) //if it is an mp3 the program is executing the next
code
    {
        //add a new attribute if the gadget it's an mp3
        memory = Float.parseFloat(theMemory.getText()); //read the text field from GUI,
        //convert it from string to float type and assigning it to variable credit
        if (memory <= 0) //check if the memory is less than 0
        {
            //pop-up an informative message for user
            JOptionPane.showMessageDialog(frame, "The MEMORY field is empty or
invalid. Please insert a positive number.");
            return; //break the program and return to the invoked method
        }
    }

```

```

        addMp3();//using addMp3 method to add new mp3 to gadget list
        Clear();//using clear method to clear the fields
    }
}

catch (NumberFormatException e) //catch the error without stopping the program
{
    //pop-up an informative message for user
    JOptionPane.showMessageDialog(frame, "An error occurred while processing the
input.");
}

}

private void addPhone()
{
    //the information to create a new gadget
    Gadget newGadget = new Mobile(model, price, weight, size, credit);
    //add the newly created Gadget object to the gadgets ArrayList
    gadgets.add(newGadget);
    System.out.println("+1 Phone added");
    System.out.println("Model: " + model);
    System.out.println("-----");
    System.out.println(numberOfGadgets() + " Gadgets in library");
    System.out.println();
}

private void addMp3()
{
    //the information to create a new gadget
    Gadget newGadget = new Mp3(model, price, weight, size, memory);
    //add the newly created Gadget object to the gadgets ArrayList
    gadgets.add(newGadget);
    System.out.println("+1 MP3 added");
    System.out.println("Model: " + model);
    System.out.println("-----");
    System.out.println(numberOfGadgets() + " Gadgets in library");
    System.out.println();
}

public void Clear() // clear the fields method
{

```

```

// set new values to each field in the GUI
theModel.setText("");
thePrice.setText("");
theWeight.setText("");
theSize.setText("");
theCredit.setText("");
theMemory.setText("");
thePhoneNumb.setText("");
theDuration.setText("");
Download.setText("");
DisplayNumber.setText("");
}

```

private void checkDuration() // creating Cehckduration method to select: the mobile from the list and make the call

```

// the number and to check if that mobile has enoth credit to make the call
{
    try //using try method to make this function user friendly and not brake the program
    {
        phoneNumber = Integer.parseInt(thePhoneNumb.getText()); // convert the text from
the GUI to type integer and assign to a global variable
        displayNumber = Integer.parseInt(DisplayNumber.getText()); // convert the text from
the GUI to type integer and assign to a global variable
        if (displayNumber >= 0 && displayNumber < gadgets.size()) //add condition to
check if the dispaly number was entered correctly
        {
            Gadget gadget = gadgets.get(displayNumber); //serchig for the gadget index in the
list
            if (gadget instanceof Mobile) // checking if the gadget of that index belongs to
mobile devices
            {
                Mobile mobile = (Mobile) gadget; // assignig mobile of tipe gadget to mobile of
type mobile
                duration = Float.parseFloat(theDuration.getText()); // convert the text from the
GUI to type float and assign to a global variable
                float currCredit = mobile.getCredit(); // creating a new local variable and asign
to the method get credit from mobile class
                if (duration >= 0 && duration <= currCredit) // checking if the duration is less
than the existing credit of the chosen phone from the gadget list
                {

```

```

        System.out.println("Your initial credit on the phone " + mobile.showModel()
+ " is: " + currCredit); // printing the initial credit
        currCredit -= duration; //subtract duration from initial credit
        mobile.setCredit(currCredit); // assign new value after the subtraction
        System.out.println("Making a call from model " + mobile.showModel() + " to
number: " + phoneNumber ); // print a message that we are calling a chosen number
        System.out.println("...");
        System.out.println("...");
        System.out.println("...");
        System.out.println("Your credit on the phone " + mobile.showModel() + " is
now: " + currCredit ); // display the remaining credit
        System.out.println("-----");
    }
    else
    {
        JOptionPane.showMessageDialog(frame, "Try another device or reduce the
calling time!"); // popping a message if its not enoth credit
    }
}
else
{
    JOptionPane.showMessageDialog(frame, "The gadget at index " +
displayNumber + " is not a mobile device.");
    System.out.println("The gadget at index " + displayNumber + " is not a mobile
device."); // popping a message if chosen device is not an mobile phone
    System.out.println("-----");
    System.out.println();
}
}
else
{
    JOptionPane.showMessageDialog(frame, "Invalid mobile index."); // popping a
message if the index device is not in the list
}
}
catch (NumberFormatException e)
{
    JOptionPane.showMessageDialog(frame, "An error occurred while processing the
input type an integer number please."); // // popping a message if the text fiel is empty
}
}
}

```

```

private void checkMemory() // creating CehckMemory method to select: the Mp3 from the
list and download if that Mp3 has enough memory to dawnload music
{
    try //using try method to make this function user friendly and not brake the program
    {
        displayNumber = Integer.parseInt(DisplayNumber.getText()); // convert the text from
the GUI to type integer and assign to a global variable
        if (displayNumber >= 0 && displayNumber < gadgets.size()) //add condition to
check if the dispaly number was entered correctly
        {
            Gadget gadget = gadgets.get(displayNumber); //serchig for the gadget index in the
list
            if (gadget instanceof Mp3)
            {
                Mp3 Mp3 = (Mp3) gadget;
                download = Float.parseFloat(Download.getText());
                float currMemory = Mp3.getMemory();
                if (download >= 0 && download <= currMemory)
                {
                    System.out.println("Your available memory on the Mp3 " +
Mp3.showModel() + " is: " + currMemory); // printing the initial memory
                    currMemory -= download; //subtract download memory from initial memory
                    Mp3.setMemory(currMemory); // assign new value after the subtraction
                    System.out.println("Download music on " + Mp3.showModel()); // print a
message that we are downloading music
                    System.out.println("...");
                    System.out.println("Remaning memory on the Mp3 " + Mp3.showModel() + "
is now: " + currMemory ); // display the remaining memory
                    System.out.println("-----");
                }
                else
                {
                    JOptionPane.showMessageDialog(frame, "Try another device or reduce the
download memory!"); // popping a message if its not enoth memory
                }
            }
            else
            {
                System.out.println("The gadget at index " + displayNumber + " is not a Mp3
device."); // popping a message if chosen device is not an Mp3
            }
        }
    }
}

```



```

        System.out.println("-----");
        System.out.println();
    }
}
else
{
    JOptionPane.showMessageDialog(frame, "Invalid Mp3 index."); // popping a
message if the index device is not in the list
}
}
catch (NumberFormatException e)
{
    JOptionPane.showMessageDialog(frame, "An error occurred while processing the
input type an integer number please."); // // popping a message if the text field is empty
}
}

public static void main(String[] args) // creating main class to open the app from cmd
{
    EShopGUI EShopGUI = new EShopGUI(); //creating a new object EShopGUI of type
EShopGUI
}
}

```

## II. Gadget CLASS

```
public class Gadget
{
    // Create 4 instances for the class gadget
    private String model;
    private double price;
    private int weight;
    private String size;

    public Gadget(String aModel,double aPrice,int aWeight,String aSize)
    //This is a constructor method for the Gadget class
    {
        // initialise instance variables
        model = aModel;
        price = aPrice;
        weight = aWeight;
        size = aSize;
    }

    //Retuning each value separately
    public String showModel()
    {
        return model;
    }

    //Print the gadget attributes list
    public void printGadget()
    {
        System.out.println("Model: " + model);
        System.out.println("Price: " + price + " £");
        System.out.println("Weight: " + weight + " grams");
        System.out.println("Size: " + size);
    }
}
```

### III. Mobile CLASS

```
public class Mobile extends Gadget
{
    private float credit;
    public Mobile(String aModel, double aPrice, int aWeight,
String aSize,float aCredit)
    {
        /* initialise instance variables for mobile class and
        adding the variables from the super class gadget*/
        super(aModel, aPrice, aWeight,aSize );
        credit = aCredit;
    }

    public void displayMobAtributes()
    {
        System.out.println("The Mobile Characteristics are:");
        super.printGadget();
        System.out.println("Your balance is: " + credit + " min");
        System.out.println("");
    }

    public float getCredit()
    {
        return credit;
    }

    public void setCredit(float aCredit)
    {
        credit = aCredit;
    }
}
```

#### IV. Mp3 CLASS

```
public class Mp3 extends Gadget
{
    // instance variables - replace the example below with your own
    private float memory;

    public Mp3(String aModel, double aPrice, int aWeight,
String aSize,float aMemory)
    {
        /* initialise instance variables for Mp3 class and
        adding the variables from the super class gadget*/
        super(aModel, aPrice, aWeight,aSize );
        memory = aMemory;
    }

    //Printing the atributes for Mp3
    public void displayMp3Atributes()
    {
        System.out.println("The Mp3 Characteristics are:");
        super.printGadget();
        System.out.println("The available space is: " + memory + " GB");
        System.out.println();
    }

    public float getMemory()
    {
        return memory;
    }

    public void setMemory(float aMemory)
    {
        memory = aMemory;
    }
}
```

## 10. References

Barnes, D. J. & Kölling, M., 2017. *Objects first with Java: a practical introduction using BlueJ*. Kent: Pearson.

beginners, J. c. f., n.d. *TUTORIAL Learn Java*. [Online]  
Available at: <https://my-learning.w3schools.com/tutorial/java>  
[Accessed 03 02 2024].

<https://app.diagrams.net/>, n.d. <https://app.diagrams.net/>. [Online]  
Available at: <https://app.diagrams.net/>  
[Accessed 10 04 2024].

Kölling, D. J. B. a. M., 2015. *objectsfirstwithjava*. [Online]  
Available at: <https://www.youtube.com/@objectsfirstwithjava>  
[Accessed 10 02 2024].