Methods
Button Creation

```
* This method initializes all the child components.
*/
private void initializeComponents() {

   this.modelLabel = new JLabel("Model:");
   this.modelTextField = new JTextField();

   this.priceLabel = new JLabel("Price:");
   this.priceTextField = new JTextField();

   this.weightLabel = new JLabel("Weight:");
   this.weightTextField = new JTextField();

   this.sizeLabel = new JLabel("Size:");
   this.sizeTextField = new JTextField();

   this.initialCreditLabel = new JLabel("Credit:");
   this.initialCreditTextField = new JTextField();

   this.initialMemoryLabel = new JLabel("Memory:");
   this.initialMemoryTextField = new JTextField();

   this.phoneNumberLabel = new JLabel("Phone Num:");
   this.phoneNumberTextField = new JTextField();

   this.durationLabel = new JLabel("Duration:");
   this.durationTextField = new JTextField();

   this.downloadSizeLabel = new JLabel("Download:");
   this.downloadSizeTextField = new JTextField();

   this.displayNumberLabel = new JLabel("Display Number:");
   this.displayNumberTextField = new JTextField();

   // Initializing buttons
   this.addMobileButton = new JButton("Add Mobile");
   this.addMobileButton.addActionListener(e -> addMobileActionHandler());

   this.addMP3Button = new JButton("Add MP3");
   this.addMP3Button.addActionListener(e -> addMP3ActionHandler());

   this.clearButton = new JButton("Clear");
   this.clearButton.addActionListener(e -> clearActionListener());
```

```
    this.displayAllButton = new JButton("Display All");
    this.displayAllButton.addActionListener(e -> displayAllActionListener());

    this.makeCallButton = new JButton("Make A Call");
    this.makeCallButton.addActionListener(e -> makeCallActionListener());

    this.downloadMusicButton = new JButton("Download Music");
    this.downloadMusicButton.addActionListener(e ->
downloadMusicActionListener());

    // Clearing all the fields.
    clearActionListener();
}
```

Adding the mobile

```
/**
* Handles a click on the add mobile button.
*/
private void addMobileActionHandler() {
```

Adding the mp3 and appropriate messages are printed if certain variables are not met

```
/**
* This Method is called when the user clicks the add mp3 button.
*/
private void addMP3ActionHandler() {
    if (this.modelTextField.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Invalid Model!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    else if (this.priceTextField.getText().isEmpty() ||
!isValidNum(this.priceTextField.getText())) {
        JOptionPane.showMessageDialog(null, "Invalid Price!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    else if (this.weightTextField.getText().isEmpty() ||
!isValidNum(this.weightTextField.getText())) {
        JOptionPane.showMessageDialog(null, "Invalid Weight!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    else if (this.sizeTextField.getText().isEmpty()) {
        JOptionPane.showMessageDialog(null, "Invalid Size!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
```

```
   else if (this.initialMemoryTextField.getText().isEmpty() ||
!isValidNum(this.initialMemoryTextField.getText())) {
       JOptionPane.showMessageDialog(null, "Invalid Memory!", "Error",
JOptionPane.ERROR_MESSAGE);
   }
   else {
       // We reach here only if all values are valid. So a MP3 is added.
       this.gadgetList.add(new Mp3(
               modelTextField.getText(),
               Double.parseDouble(priceTextField.getText()),
               Integer.parseInt(weightTextField.getText()),
               sizeTextField.getText(),
               Integer.parseInt(initialMemoryTextField.getText())
       ));


this.displayNumberTextField.setText(String.valueOf(this.gadgetList.size() -
1));

       JOptionPane.showMessageDialog(null, "Success! MP3 Saved", "Success",
JOptionPane.INFORMATION_MESSAGE);
   }
}
```

When the clear button is clicked

```
* This method is called when the clear button is clicked.
*/
private void clearActionListener() {

   // Clearing all the fields.
   this.modelTextField.setText("");
   this.priceTextField.setText("");
   this.weightTextField.setText("");
   this.sizeTextField.setText("");
   this.initialCreditTextField.setText("");
   this.initialMemoryTextField.setText("");
   this.phoneNumberTextField.setText("");
   this.durationTextField.setText("");
   this.downloadSizeTextField.setText("");
   this.displayNumberTextField.setText("-1");
}
```

When the display all button is clicked

```
 private void displayAllActionListener() {
```

```
   String[] columnNames = {"Model", "Price", "Weight", "Size", "Credit",
"Memory"};
   Object[][] data = new Object[gadgetList.size()][columnNames.length];

   for (int i = 0; i < this.gadgetList.size(); i++) {
       Gadget gadget = gadgetList.get(i);

       data[i][0] = gadget.getModel();
       data[i][1] = gadget.getPrice();
       data[i][2] = gadget.getWeight();
       data[i][3] = gadget.getSize();

       if (gadget instanceof Mobile) {
           Mobile mob = (Mobile) gadget;
           data[i][4] = mob.getCreditRemaining();
       }
       else {
           data[i][4] = "--";
       }

       if (gadget instanceof Mp3) {
           Mp3 mob = (Mp3) gadget;
           data[i][5] = mob.getMemoryAvailable();
       }
       else {
           data[i][5] = "--";
       }
   }
```

When the make call button is clicked

```
/**
* This method is called when the make call button is clicked.
*/
private void makeCallActionListener() {

   if (this.phoneNumberTextField.getText().isEmpty()) {
       JOptionPane.showMessageDialog(null, "Invalid Phone Number!", "Error",
JOptionPane.ERROR_MESSAGE);
   }
   else if (this.durationTextField.getText().isEmpty() ||
!isValidNum(this.durationTextField.getText())) {
```

```
        JOptionPane.showMessageDialog(null, "Invalid Duration!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    else if (this.displayNumberTextField.getText().isEmpty() ||
!isValidNum(this.displayNumberTextField.getText())) {
        JOptionPane.showMessageDialog(null, "Invalid Display Number!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
    else {
        int selectedGadget =
Integer.parseInt(this.displayNumberTextField.getText());

        if (selectedGadget < 0 || selectedGadget >= this.gadgetList.size()) {
            // Falls out of range.
            JOptionPane.showMessageDialog(null, "The selected gadget index is
out of range!", "Error", JOptionPane.ERROR_MESSAGE);
        }
        else {
            Gadget gadget = this.gadgetList.get(selectedGadget);
            if (gadget instanceof Mobile) {
                // It is a Mobile so making the call.
                Mobile mob = (Mobile) gadget;
                String response = mob.call(phoneNumberTextField.getText(),
Integer.parseInt(durationTextField.getText()));

                JOptionPane.showMessageDialog(null, response, "Information",
JOptionPane.INFORMATION_MESSAGE);
            }
            else {
                JOptionPane.showMessageDialog(null, "The selected gadget is an
MP3. It doesn't support calls.", "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```

When the download music button is clicked

```
/**
* This method is called when the download music button is clicked.
*/
private void downloadMusicActionListener() {
    if (this.downloadSizeTextField.getText().isEmpty() ||
!isValidNum(this.downloadSizeTextField.getText())) {
        JOptionPane.showMessageDialog(null, "Invalid Download Size!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
```

```
   else if (this.displayNumberTextField.getText().isEmpty() ||
!isValidNum(this.displayNumberTextField.getText())) {
       JOptionPane.showMessageDialog(null, "Invalid Display Number!", "Error",
JOptionPane.ERROR_MESSAGE);
   }
   else {
       int selectedGadget =
Integer.parseInt(this.displayNumberTextField.getText());

       if (selectedGadget < 0 || selectedGadget >= this.gadgetList.size()) {
           // Falls out of range.
           JOptionPane.showMessageDialog(null, "The selected gadget index is
out of range!", "Error", JOptionPane.ERROR_MESSAGE);
       }
       else {
           Gadget gadget = this.gadgetList.get(selectedGadget);
           if (gadget instanceof Mp3) {
               // It is a Mobile so making the call.
               Mp3 mp3 = (Mp3) gadget;
               String response =
mp3.downloadMusic(Integer.parseInt(downloadSizeTextField.getText()));

               JOptionPane.showMessageDialog(null, response, "Information",
JOptionPane.INFORMATION_MESSAGE);
           }
           else {
               JOptionPane.showMessageDialog(null, "The selected gadget is a
Mobile. It doesn't support downloads.", "Error", JOptionPane.ERROR_MESSAGE);
           }
       }
   }
}
```

This checks if the field has a valid number

```
/**
* This function checks if the field has a valid number.
* @param text The text to be checked.
* @return True if the text is a valid number.
*/
private boolean isValidNum(String text) {

   try {
       Double.parseDouble(text);
       return true;
   }
```

```
    catch (NumberFormatException ex) {
        return false;
    }
}
```

Action Performed

```
public int getCreditRemaining() {
    return creditRemaining;
}
```

This returns the int value of creditRemaining
Adding credit to the mobile.

```
System.out.println("Error! Insufficient credits remaining. Cannot make this
call. "
        + "Remaining Credit: " + this.creditRemaining);
```

This prints out a message if the credit is too low to make the call.

```
System.out.println("Success! Phone Number: " + phoneNumber
        + ", Duration: " + duration
        + ", Remaining Credit: " + this.creditRemaining);
```

This prints out a message if the credit was accepted as there is just enough or surplus of it and it will make the call and will also show how many remaining credit is left in the balance.

```
// User does not have enough memory for this download.
System.out.println("Error! Insufficient memory remaining. Cannot download. "
        + "Available Memory: " + this.memoryAvailable);
```

This is printed out when there isn't enough memory to download the music and will show how much memory is available for that gadget.

```
    // User has enough memory for this download.
        this.memoryAvailable -= memoryRequired;
        System.out.println("Success! Available Memory" + this.memoryAvailable);
```

```
        return "Success! Available Memory: " + this.memoryAvailable;
    }
}
```

This prints out a message stating the download was valid and says how much remaining memory is in that gadget.