### lmu_blakCourse Submission Cover Sheet Module: CC4001 Programming Engineering

### Component no: 003

### Weighting: 60% of module mark

### Deadline: 1st of May 2024

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

PLAGIARISM

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

(Signature:) Date:

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

Extracts from University *Regulations on*

Cheating, Plagiarism and Collusion

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

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

Some notes on what this means for students:

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

1

**Assignment**

This assignment will be marked out of 100 and carries 60% of the overall module weighting. Your .java files and report for this part must be uploaded to WebLearn and submitted by 3pm on Friday 1st of May 2024. The assignment must be carried out individually so you must not obtain help from anyone other than the module teaching staff. You must not copy code from any source apart from the module core text and the module WebLearn site. Collusion, plagiarism (unreferenced copying) and other forms of cheating constitute Academic Misconduct, which can lead to failure of the module and suspension from the University.

**Aim**

The aim of this assignment is to create 4 classes. The main class will represent a gadget, together with two subclasses to represent a mobile phone and an MP3 player respectively. You then have to make a graphical user interface class (GUI) for a system that stores details of gadgets in a shop. The class will contain a main method and will be tested using the command prompt. You will also need to write a report about your program. You are advised to study the learning aid on reflective practice, which is linked from the module WebLearn, and the learning aids entitled “Using the Command Prompt” and “Examples of screenshots for Coursework” which are included in the learning and teaching materials. Your report should be no more than 2500 words in length (excluding the table of contents, class diagram, pseudocode and code listing).

**Deliverables**

When you are ready to submit your solution, upload your Gadget.java, Mobile.java and MP3.java and GadgetShop.java file to the GitHub. Provide a link to your CS4001-CW-repository in the first page of your assignment. You will also have to upload your word report to the WebLearn when you are absolutely sure that you have uploaded the correct files, press the submit button. You can include you programme codes in the appendix of your report. (ask your tutor if you need help completing those classes to the specification of the first part of the coursework.)

**Program**

Program

The program should consist of the following classes (with no additional attributes or methods).

1. The Gadget class has four attributes, which correspond to the model, the price (in pounds), the weight (in grams), the size (e.g. “71mm x 137mm x 9mm”) respectively. The price is a decimal number, the weight is an integer, and the model and the size are strings of text. Each attribute is initialised in the constructor, by being assigned the value of one of the constructor's four parameters, and each attribute has a corresponding accessor method. A display method will output (suitably annotated) the model, the price, the weight and the size
2. The Mobile class is a subclass of the Gadget class and has just one attribute, which corresponds to the (whole) number of minutes of calling credit remaining. The attribute is initialised in the constructor by being assigned the value of one of the constructor's five parameters and it has a corresponding accessor method. The other parameters of the constructor represent the model the price, the weight and the size of the mobile phone and these four values are passed to the constructor of the Gadget class.The Mobile class has a method so that the user of the mobile can add (via a parameter) calling credit to the amount that they currently have. If the amount of credit provided by the user is greater than zero then this amount is added to the amount of credit stored in the mobile. Otherwise, a message is displayed prompting the user to enter a positive amount. There is also a method to represent the user making a phone call. The user needs to provide the phone number and the duration of the call in minutes. If there is enough credit then a message giving the phone number and duration is displayed and the remaining calling credit is reduced by the number of minutes that the call lasted. Otherwise, a message informing the user that there is insufficient credit to make the call is displayed. A method to display the details of the mobile is required. It must have the same signature as the display method in the Gadget class. It will call the method in the Gadget class to display the model, the price, the weight and the size. The number of minutes of calling credit remaining is then output suitably annotated.
3. The MP3 class is also a subclass of the Gadget class and has just one attribute, which corresponds to the available memory. The attribute is initialised in the constructor by being assigned the value of one of the constructor's five parameters and it has a corresponding accessor method. The other parameters of the constructor represent the model, the price, the weight and the size of the MP player and these four values are passed to the constructor of the Gadget class.The MP3 class also has a method for downloading music which takes a parameter representing the amount of memory that the music will take up and, if there is sufficient available memory on the MP3 player, decreases the available memory accordingly, otherwise an appropriate error message is printed. There is also a method for deleting music which takes a parameter representing the amount of memory that the music took up and increases the available memory of the MP3 player accordingly.A method to display the details of the MP3 player is required. It must have the same signature as the display method in the Gadget class. It will call the method in the Gadget class to display the model, the price, the weight and the size. The available memory is then output suitably annotated.
4. The GadgetShop class should store an array list (not an array) of type Gadget to hold the gadgets. An example GUI is shown below: Your GUI (GadgetShop class) should contain the same components, but you are free to use a different layout if you feel that it improves the aesthetics, ease of use etc.



**Text Field Input Methods**

For each text field, write a method to read its contents and return the value. For the model, size and phone number, the value should simply be returned as a string but, for each of the other text fields, the input value should be converted into its correct numerical data type and then that value should be returned.

There should be text fields for entering:

* The model
* The price
* The weight
* The size
* The initial credit (when adding a mobile phone)
* The initial available memory (when adding an MP3 player)
* The phone number (when making a mobile phone call)
* The duration (when making a mobile phone call)
* The download size (when downloading music to the MP3 player)
* The display number, which is the same as the index of the gadget in the array list

The input method for the display number is different from the input methods for the other text fields. The method first initialises the display number to -1. If the input from the user via the GUI is valid, the display number will be changed to a value that corresponds to a gadget in the array list. The input from the display number text field must therefore be tested using a try/catch statement to ensure that the gadget number entered is an integer and, if it is, that it is in the correct range. If the value entered is an integer but is not in the correct range, a suitable error message is displayed using a message dialog box. If the display number entered is not an integer then an alternative error message is displayed using a message dialog box. The method will therefore either return –1 or a value that corresponds to a gadget in the array list, depending on whether or not there was an error in the input. Any method that gets the display number should check its value and only use it if its value is not equal to –1.

**Buttons**

The GUI should have the following buttons:

1. **Add Mobile**

The model, price, weight, size and credit are input via the GUI. When this button is pressed, the methods to get the model, price, weight, size and credit are called, and these values are used to create a new object of type Mobile, which is added to the array list of gadgets.

1. **Add MP3**

The model, price, weight, size and memory are input via the GUI. When this button is pressed, the methods to get the model, price, weight, size and memory are called, and these values are used to create a new object of type MP3, which is added to the array list of gadgets.

1. **Clear**

When this button is pressed, the text is cleared from each of the ten text fields.

1. **Display All**

When this button is pressed, the display number is printed and the method to display the details of the gadget is called, for each gadget in the array list.

1. **Make A Call**

The display number, phone number and duration are input via the GUI. When this button is pressed, the method to get the display number is called and its value is checked. If it is not equal to –1:

* The display number is used to get the gadget from the array list and cast it to Mobile.
* The method to make a call in the Mobile class is called with the phone number and duration entered.

1. **Download Music**

The display number and download size are input via the GUI. When this button is pressed, the method to get the display number is called and its value is checked. If it is not equal to –1:

* The display number is used to get the gadget from the array list and cast it to MP3.
* The method to download music in the MP3 class is called with the download size entered.

**Marks**

Marks will be awarded as follows:

1. GitHub Link
2. A class diagram of all 4 classes
3. GUI
4. The actionPerformed method
5. The main method
6. Functionality of buttons
7. Reading input
8. Use of try/catch, checking input and displaying appropriate message dialogs
9. Program style (see http://www.bluej.org/objects-first/styleguide.html)

**Report**

The report should contain:

1. The GitHub link to your CW repository **(5%)**
2. A class diagram including GadgetShop class and other 3 classes showing the name of the class, the data types and names of the fields, and the return types and signatures of all of the methods. **(5%)**
3. A short description of each of your methods in Gadget.java, Mobile.java and MP3.java **(9%)**
4. Pseudocode for the following button-handling methods: **(18%)**

* Getting the display number from the GUI
* Adding a mobile
* Adding an MP3
* Displaying all gadgets in the array list
* Making a call
* Downloading music

1. Textboxes, input check with try/catch **(9%)**
2. Buttons and Action Performed Methods **(14%)**
3. Test that the programme can be run in command prompt **(3%)**
4. GUI: display, add mobile, add MP3, display, make a call, download music **(10%)**
5. You should give evidence (through appropriate screenshots) of the following testing that you carried out on your program: **(7%)**

Test 1: Adding a mobile to the array list

Test 2: Adding an MP3 player to the array list

Test 3: Displaying the details of all of the gadgets in the array list

Test 4: Making a call

Test 5: Downloading music

Test 6: Test that the program can be compiled and run using the command prompt, including a screenshot similar to Figure 1 in the command prompt learning aid.

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

(Include a screenshot of the dialog box, together with a corresponding screenshot of the GUI, showing the values that were entered.)

1. The report should contain a section on error detection and error correction where you give examples and evidence of three errors encountered in your implementation. The errors (syntax and/or runtime) should be distinctive and not of the same type. **(5%)**
2. The report should contain a conclusion, where you evaluate your work, reflecting on what you learnt from assignment. The report should include a title page, a table of contents (with page numbers), and a listing of the code (in an appendix). Marks will also be awarded for the quality of writing and the presentation of the report. Good program style, particularly naming, layout and documentation. See http://www.bluej.org/objects-first/styleguide.html for details. **(10%)**
3. Additional marks will be awarded for new features **(5%)**

|  |  |
| --- | --- |
| **A picture containing text  Description automatically generated**  **School of Computing and Digital Media**  **Marking Criteria for Coursework** | |
| **Marking criteria** | **Mark range** |
| An exceptional submission, with extensive and detailed knowledge based on a high level of additional background research; a high degree of critical analysis, evaluation and original insight; excellent organization and presentation. | 90 – 100 |
| In addition to the requirements for grades of 70-79% (below), an outstanding submission incorporating a high level of originality, depth and critical insight and going well beyond expected work. | 80 – 89 |
| An excellent submission, going beyond expected work; commanding understanding and appreciation of the central points; well-written and effectively structured; evidence-based, critical and logical analysis; comprehensive and correct referencing of sources. | 70 – 79 |
| A very good and comprehensive submission that fulfills the assignment brief; relevance and accuracy; clear structure and evidence-based; a sound grasp of the subject and ability to think about it effectively and critically; correct referencing of sources. | 60 – 69 |
| A solid submission that fulfills most of the assignment brief; adequate structure; mostly accurate, with few errors or omissions; some limitations in scope, critical thinking and argument; a consistent attempt at referencing sources. | 50 – 59 |
| A basic but incomplete submission, with limited relevant information; lacks logical and coherent structure, with some significant errors or omissions; contains sparse and/or irrelevant information and lacks an evidence-based approach; some limitations with referencing of sources; poor grammar and spelling. | 40 – 49 |
| An unsatisfactory submission, with some substantial errors, omissions or irrelevancies; barely acceptable amount of information relevant to the question; poor structure, presentation and expression and referencing of sources. | 30 – 39 |
| An unsatisfactory submission, that doesn’t meet the learning outcomes with many substantial errors, misconceptions, omissions or irrelevancies; little information relevant to the question; very poor structure, presentation and expression. | 1 – 29 |
| Non-submission or submission of work which cannot be given any credit (e.g., blank submission, incorrect assignment). | 0 |