香港公開大學
THE OPEN UNIVERSITY OF HONG KONG

COMPS203F
Marker No.:
Total Mark:

2021 Spring Examination (UG)

INTERMEDIATE JAVA PROGRAMMING AND USER INTERFACE DESIGN (2021 Spring Term)

27 May 2021 **Time Allowed: 2 hours** **14:00–16:00**

Student Number								

THIS IS AN ONLINE OPEN-BOOK EXAMINATION

1. This examination paper is available on OLE Online Exam Paper(s) page.
2. You should answer the examination paper using **your OWN efforts only**. Any plagiarism behaviour discovered will have serious consequences, including getting zero mark for this examination.
3. This is an open-book examination. You can read books and access the Internet. However, you cannot copy answers from any source.
4. Use your own paper to answer the questions of this examination paper in English. You may optionally print our answer book (on OLE) for writing your answers. **Write your 8-digit student ID and name on your answer sheets.**
5. You can start as soon as you successfully download the question paper.
6. Read the instructions in the examination paper carefully and **write** the question numbers and answers clearly. It may not be possible to award marks where the writing is very difficult to read. **You are required to WRITE your answers and typed answers are NOT allowed** (unless prior approval is given)**.**
7. At the end of the examination, scan your answers and check each page. **Make sure the image of each page is sharp enough** to be seen. Generate and upload a PDF file containing your answers to OLE "Online Exam". Normally only this OLE version will be marked.
8. As a backup, attach your PDF file in an email to kwlee@**study**.ouhk.edu.hk and twong@ouhk.edu.hk with subject "COMPS203F Exam: 11223344" (where 11223344 is your student ID). Remember to CC your email address a copy.

   Another backup email address is comps203f@live.ouhk.edu.hk, If either (1) OLE upload or (2) sending email to the above email addresses is not successful, send a copy of your answers to this email address. If both (1) and (2) are fine, this one is not needed.

PART I (60 marks)

(i) You should attempt **ALL** *the questions in this part of the examination paper.*
(ii) There are altogether ***six*** *questions in this part.*
(iii) You are advised to spend one hour and ten minutes on this part.
(iv) Write all your answers in your answer paper.

Question 1

The first few lines of a class `LuckyDraw` are shown below:

```
public class LuckyDraw extends JFrame implements ActionListener {
  private JLabel label = new JLabel("Lucky Numbers: ");
  private JButton button = new JButton("Draw 3");
```

The GUI includes a label and a button. The layout manager is `FlowLayout` manager. After the button is pressed, lucky numbers are shown on a frame similar to the one on the right side.

(a) Write the constructor `LuckyDraw()` to generate the GUI (with label containing "Lucky numbers: ") and add the object itself as the action listener of the button. When the window is closed, the program should terminate.

[3 marks]

(b) When the button is pressed, 3 random integers from 1 to 10 (inclusive) is drawn **without duplication** and displayed on the label with a space separating two integers as shown in the above screenshot. Write the method `actionPerformed(ActionEvent ae)` to do this action.

[7 marks]

Question 2

Fill in the blanks using the most suitable word(s) given below:

zero, one, two, ten, unlimited, instance, class, superclass, subclass, method, attribute, abstract, concrete, information hiding, inheritance, overloading, overriding, dynamic binding.

Capitalize the first letter where appropriate.

(a) In Java, the maximum number of direct superclass(es) for a class is: _____.

[2 marks]

(b) Using the access modifier "private" in Java is to achieve _____.

[2 marks]

(c) In a class, how many methods can have the method name "main"? _____

[2 marks]

(d) If the methods of ClassB can override the methods of ClassA, ClassB is a(n) _____ of ClassA.

[2 marks]

(e) _____ methods have both method headings and bodies.

[2 marks]

Question 3

An attribute of a class `ObjectFile` is declared as below:

```
private List<Student> studentList = new ArrayList<>();
```

(a) Assuming that the class `Student` has been written correctly, write a method `save(String fileName)` to save `studentList` to a file with file name `fileName`. Handle possible exception(s) using `try-catch` and output a suitable message using the `getMessage()` method if an exception occurs.

[4 marks]

(b) Write another method `read(String fileName)` to read an `ArrayList` of `Student` objects from a file with file name `fileName` and assign the list to `studentList`. Handle possible exception(s) using `try-catch` and output a suitable message using the `getMessage()` method if an exception occurs.

[5 marks]

(c) Write the class `Student` up to and including the first `"{"`. This is (usually) the first line of the class and you don't need to write other code of the class.

[1 mark]

Question 4

The information of phones for phone shop A and phone shop B is to be stored in two maps (`Map<String, Double>`) referenced by `phoneShopA` and `phoneShopB` respectively. The model number of the phone is the key and it is assumed to be unique in each map. The price of the phone is the value.

(a) Write a method `meanPrice(Map<String, Double> phoneShop, double bound)` with an **enhanced** for loop to find and return the average price of phones with prices not more than `bound` in `phoneShop`.

[5 marks]

(b) Assume there are many entries in the maps and one phone shop can have some phones not available in the other. Write a method `lowerPrice()` with an enhanced for loop to print out the model number of each phone available in **both** of the phone shops and its lower price.

[5 marks]

Question 5

(a) Write an **abstract** class `Staff` with an **abstract** method `findSalary()`, which, when implemented, returns the salary of the staff as a real number.

[2 marks]

(b) Write a subclass `FullTimeStaff` of `Staff` with a private real number attribute `salary`. Add a constructor to initialize the attribute with its parameter. Also write any method(s) of `Staff` and/or `FullTimeStaff` needed.

[4 marks]

(c) Write a subclass `PartTimeStaff` of `Staff` with a private integer attribute `hoursPerMonth` and real number attribute `hourlyRate`. Also write a constructor `PartTimeStaff(int hoursPerMonth, double hourlyRate)` which suitably initializes the attributes. The salary of `PartTimeStaff` is the product of `hoursPerMonth` and `hourlyRate`. Also write any method(s) of `Staff` and/or `PartTimeStaff` needed.

[4 marks]

Question 6

(a) A table of a database has been created using

```
create table computer (id varchar(10), price numeric(6,1));
```

Write a Java method `insertComputer(Connection conn, String id, double price)` to insert the computer information into the table using `conn`, which is ready to be used. Use `try-catch` to handle the exception(s) but no `import` statements are needed to be written.

[5 marks]

(b) Write a Java method `createJSON()` to return an object with type `JSONObject` which contains the following content :

```
{ "shop": {
    "computer": [
      { "id": "AB 123",
        "price": 2388.8
      }
    ]
  }
}
```

No `import` statements, no formatting, and no `try-catch` are needed.

[5 marks]

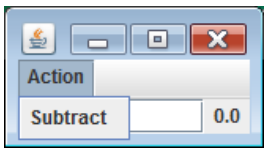
[END OF PART I]

PART II (40 marks)

(i) You should attempt **ALL** *questions. Each question is worth 20 marks.*
(ii) Show all your work steps.
(iii) You are advised to spend fifty minutes on this part.
(iv) Write all your answers in your answer paper.

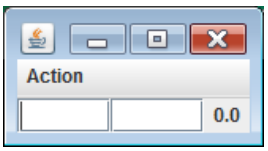
Question 7

(a) Write a class `Calculator`, which is a subclass of `JFrame`, to create a frame with a pull-down menu (containing an "Action" menu and a "Subtract" menu item) as shown on the right. Remember to include the `import` statement(s). There is no need to implement other parts of the GUI and handle actions yet.



[6 marks]

(b) State the modifications in part (a) to implement other parts of the GUI which include two new text fields and a label containing 0.0. `BorderLayout` manager is used and the lengths of the text fields are 10. There is also no need to handle actions at this moment. You only need to write the modifications and they are assumed to be in proper places.



[3 marks]

(c) State the modifications in (b) to handle actions. When the menu item is chosen, the text on the label is replaced by the result of subtracting the value in the right text field from that in the left text field. Use exception handling to check if the numbers are valid real numbers. If either one is invalid, display a message dialog box with message "Please use valid real number(s)" on top of the original window. Remember to include any additional `import` statement(s). Hint: `NumberFormatException` may occur during conversion if the real number input is not valid.

[11 marks]

Question 8

Dickson has come up with a way to encode text with repeating patterns and the following gives the definition of his "encoded string":

- a single lower-case letter is an encoded string
- (e_1 e_2 ... e_t n) is also an encoded string, where t and n are positive integers and e_i ($1 <= i <= t$) is an encoded string. The first and last characters are "(" and ")" respectively. Two consecutive items e_i and e_{i+1} (also e_t and n) are separated by exactly one space.

Observe that an encoded string of one character is the same as a decoded string. To decode the string (e_1, e_2,.., e_t n), we decode each e_i, concatenate those decoded strings into a new string, and concatenating n copies of new string. For example:

- `x` would be decoded as `x`,
- `(t 3)` would be decoded as `ttt`,
- `(b c 2)` would be decoded as `bcbc`,
- `(a (b c 2) 3)` would be decoded as `abcbcabcbcabcbc`.

(a) Write a class method `simpleDecode(String str)` which decodes `str` and returns the original string. You can assume the string `str` contains **at most** one pair of parentheses.

[8 marks]

(b) Write a class method `decode(String str)` which decodes `str` and returns the original string. The string `str` can contain **any number** of pairs of parentheses.

[12 marks]

Appendix: Concise Java Statement Examples and Partial Method List

This appendix is provided to reduce the load of memorizing the syntax and methods learnt. This is not a complete reference and total correctness is not guaranteed. Some methods not listed here need to be used.

Statement Examples

```
int a, b, c, e;
int[] f = new int[9];
double d;
TicketCounter tc = new TicketCount();
if (a == b && b != 1 || c <= 0) {
  d = 0;
} else {
  d = 1;
}
switch (e+2) {
  case 2 : case 5:
     f[4] = (int) d; break;
  default :
     tc.increase(); break;
}
```

```
public class Example {
  boolean good;
  int j=0, k=0, m=0, n=0;
  public double loops(String s) {
     for (int i=0; i<n; i++) {
        j += i;
     }
     do {
        k = k + 2;
     } while ( k < 5 );
     while (true || k > 2) {
        m++;
     }
     return (double) k;
  }
}
```

Method List

```
  Collection --    add(o), contains(o), isEmpty(), remove(o), size(), toArray()
    List --           add(i,o), get(i), indexOf(o), lastIndexOf(o), remove(i),
                      set(i,o)
    Set --            <see Collection>

  Container --     add(co), add(co, i)

  File --          exists(), File(s), isFile(), isDirectory(), length()

  InputStream --   read(), read(b[])
    FileInputStream -- FileInputStream(f), FileInputStream(s)

  JButton --       JButton(s), setText(s)
  JCheckBox --     isSelected(), JCheckBox(s), setSelected()
  JFrame --        getContentPane(), JFrame(s), pack(), setJMenuBar(mb),
                   setLayout(lm), setSize(i,j), setVisible(bo), show()
  JLabel --        JLabel(s)
  JMenuBar --      add(m), JMenuBar()
  JMenu --         add(mi), addSeparator(), JMenu(s)
  JMenuItem --     JMenuItem(s)
  JOptionPane --   showMessageDialog(o,s), showInputDialog(o,s),
                   showConfirmDialog(o,s)
  JPanel --        add(co), add(co, i)
  JRadioButton --  isSelected(), JRadioButton(s), setSelected()
  JTextField --    JTextField(i)
  JTextArea --     JTextArea(i,j)

  Map --           containsKey(k), containsValue(o), get(k), keySet(), put(k,o),
                   remove(k), values()

  OutputStream --  write(i), write(b[])
    FileOutputStream -- FileOutputStream(f), FileOutputStream(s)

  String --        charAt(i), compareTo(s), equals(o), indexOf(c), indexOf(s),
                   length(), replace(c,c), substring(i,j), toCharArray(),
                   toLowerCase(), toUpperCase()
```

where b: byte, b[]: byte array, bo: boolean, c:char, co: GUI component, f: File, i,j: int, k: key(Object), lm: layout manager, m: menu, mb: menu bar, mi: menu item, o: Object, s: String

[END OF EXAMINATION PAPER]