Xi'an Jiaotong-Liverpool University

西交利物浦大学

Paper CODE	EXAMINER	EMAIL OF EXAMINER	TEL
CPT105/2022/23 S1 Resit Exam	Yushan Pan	Yushan.Pan@xjtlu.edu.cn	+86-(0)512-89165347

1st SEMESTER 2022/23 RESIT EXAMINATION

Undergraduate – Year 1

INTRODUCTION TO PROGRAMMING IN JAVA

TIME ALLOWED: 2 HOURS

INSTRUCTIONS TO CANDIDATES

1. **This is a CLOSED-book examination. Please tick the integrity disclaimer when handing in your answer sheets. Please complete the assessment independently and honestly.**
2. **The total marks available are 100.**
3. **The exam paper consists of 14 printed pages.**
4. **This exam consists of 3 sections:**

   Section A consists of single choice questions with 13 questions worth 2 mark each for a total mark of 26.

   Section B consists of multiple-choice questions with 6 questions worth 3 marks each for a total mark of 18.

   Section C consists of coding tasks with 3 questions. The first two is worth 15 marks each. And the last one is worth 26 marks. In total 56 marks for Section C.

   Answer all questions. There is NO penalty for providing a wrong answer.
5. **Only English solutions are accepted. Answers need to be handwritten on answer sheets. Please clearly indicate the question numbers before your solutions. Please also indicate ONLY your student ID number on answer sheets.**
6. **The exam duration is 2 hours. If there are any major problems preventing you from continuing the exam or submitting your answers in time, please do not hesitate to email the Module Examiner or Assessment Team of Registry (assessment@xjtlu.edu.cn).**

Section A Single Choice (26 marks)

This section consists of 13 questions, each worth 2 marks. There is only one correct answer for the following questions.

1. Suggest a = -2, b = 3. Which Boolean statement's value is True?

A. a >= b && a >= 1

B. (a >= 0 && b <= 0) || (a < 0 && b > 0)

C. (a < 0 && b < 0) || a >= b

D. a >= 0 && b >= 0

2. According to the code fragment below, which option's return value is 'Statement 1 Statement 2 Statement 3'?

```
1. int a;
2. if (a >= 1)
3.     System.out.print("Statement 1 ");
4. else if (a >= 2)
5.     System.out.print("Statement 2 ");
6. else if (a >= 3)
7.     System.out.print("Statement 3 ");
```

A. a=0

B. a=1

C. a=2

D. a=3

3. What is the output of the following program fragment?

```
1. int count = 0;
2. for(int i = 0; i <= 10; i = i++ ) {
3.     count += i;
4. }
5. System.out.print(count);
```

A. 55

B. 30

C. 25

D. 18

4. What is the output of the following program fragment?

```
int count = 0, loop_int = 10;
while (loop_int >= 0){
    loop_int -= 1;
    count += loop_int;
}
System.out.println(count);
```

A. 45

B. 44

C. 17

D. 9

5. What is the output of the following program fragment?

```
int count = 0;
for (int i = 1; i <= 5; i++)
    for (int j = 1; j <= i; j++)
        count += i * j;
System.out.print(count);
```

A. 140

B. 275

C. 65

D. 225

6. What is the output of the following program fragment?

```
int count = 0;
int[] list = {1, 2, 3, 4, 5, 6, 7};
for (int item : list)
    if(item % 2 == 1)
        count++;
System.out.println(count);
```

A. 2

B. 3

C. 4

D. 5

7. What is the most possible output of the following program fragment?

```
int N = 10000000;
int count = 0;
double x, y;
for (int i = 0; i < N; i++) {
        x = Math.random( );
        y = Math.random( );
        if (x*x + y*y <= 1) count++;
}
System.out.println((double)count*1.0 / N );
```

A. 0

B. 1

C. 0.5

D. 0.785

8. What is the output of the following program fragment?

```
String str = "1#";
int[] list = {1, 2, 3};
try{
    System.out.print(list[3]);
    System.out.print(Integer.parseInt(str));
}
catch (ArrayIndexOutOfBoundsException e){
    System.out.print("#Exception 1;");
}
catch (NumberFormatException e){
    System.out.print("#Exception 2;");
}
```

A. 13

B. #Exception 1;#Exception 2;

C. #Exception 1;

D. #Exception 2;

9. What is the output of the following program fragment?

```
1.  public class MyData {
2.      public static int i;
3.      public static int j;
4.
5.      public MyData(int input_a, int input_b){
6.          i = input_a;
7.          j = input_b;
8.      }
9.
10.     public int returnSum(){
11.         return i + j;
12.     }
13.
14.     public static void main(String[] args) {
15.         MyData myData1 = new MyData(1, 2);
16.         MyData myData2 = new MyData(2, 3);
17.         System.out.print(myData1.returnSum() + myData2.returnSum());
18.     }
19. }
```

A. 7

B. 8

C. 9

D. 10

10. What is the output of the following program fragment?

```
1.  public class FList {
2.      public static int getValue(int index){
3.          if (index == 0)
4.              return 1;
5.          else
6.              return getValue(index - 1) * 2 + 1;
7.      }
8.
9.      public static void main(String[] args) {
10.         System.out.println(getValue(6));
11.     }
12. }
```

A. 15

B. 31

C. 63

D. 127

11. What is the output of the following program fragment?

```
1.  public class DealStr {
2.      public static String changeStr(String s){
3.          char[] temp = s.toCharArray();
4.          StringBuilder ans = new StringBuilder();
5.          for (char c : temp){
6.              if ((int)'a' <= (int)c && (int)c <= (int)'z')
7.                  ans.append(c);
8.              else
9.                  ans.append((char)((int)c + ((int)'a' - (int)'A')));
10.         }
11.         return String.valueOf(ans);
12.     }
13.
14.     public static void main(String[] args) {
15.         System.out.println(changeStr("aaAAC"));
16.     }
17. }
```

A. AAaac

B. aaAAC

C. AAAAC

D. aaaac

The following background description is used for questions 12 and 13.

Hero Game.
We are writing the hero part scripts of a game. According to the game design, there are three types of heroes in this game: Warrior, Archer, and Priest. Warriors can attack at close range, archers can attack at long range, and priests can heal their teammates. All the heroes can run.

12. According to the above Hero Game's design. Which option below is the most suitable way to implement the skill, 'run'?

A. Create an Interface to implement the skill, 'run'.

B. Create an abstract class to implement the skill, 'run'.

C. Create an Abstract class for all heroes as an introductory class and include the skill 'run' in it.

D. Create an interface for all heroes as an introductory class and include the skill 'run' in it.

13. Which method is the best to avoid the Archer class having the 'heal' skill?

A. Write comments to tell the other programmers.

B. Only implement the 'heal' skill in the Archer class

C. Implement the 'heal' skill through an abstract class

D. Implement the 'heal' skill through an interface

Section B Multiple Choice Questions (18 marks)

This section consists of six questions, each worth 3 marks. There might be one or more than one correct answers. You will get a half score if your answer is incomplete. You will get no marks if your answer consists of a wrong choice.

1. According to the code fragment below, which option's return value is 6?

```
1.  public static int checkStr(String str, char c){
2.      char[] s = str.toCharArray();
3.      int ans = 0;
4.      for (char _c : s){
5.          if (_c == c)
6.              ans++;
7.      }
8.      return ans;
9.  }
```

A. checkStr("dadbdadBdadbabACbaCbaccccAA", 'a')

B. checkStr("dadbdadBdadbabACbaCbaccccAA", 'b')

C. checkStr("dadbdadBdadbabACbaCbaccccAA", 'c')

D. checkStr("dadbdadBdadbabACbaCbaccccAA", 'd')

2. Which option's return value is true to the code fragment below?

```
1.  public static boolean ifGoodStr(String s){
2.          int l = 0, d = 0;
3.          for (int i = 0; i < s.length(); i++){
4.              if (Character.isLetter(s.charAt(i)))
5.                  l++;
6.              else
7.                  d++;
8.          }
9.          return l > d;
10.     }
```

A. ifGoodStr("a123b")

B. ifGoodStr("a123b456")

C. ifGoodStr("abcdefg")

D. ifGoodStr("abcdefg123456")

3. Which option's return value is true to the code fragment below?

```
1.  public static boolean isBalanceStr(String s){
2.          boolean _i = false, _c = false, _l = false;
3.          for (int i = 0; i < s.length(); i++){
4.              if (Character.isLetter(s.charAt(i))) _l = true;
5.              else if (Character.isDigit(s.charAt(i))) _i = true;
6.              else  _c = true;
7.          }
8.          return _i && !_c && _l;
9.  }
```

A. isBalanceStr("a123b")

B. isBalanceStr("a1@3")

C. isBalanceStr("a1 3b")

D. isBalanceStr("aC23b")

4. Which option's return value is true to the code fragment below?

```
1.  public static boolean isBRStr(String s){
2.      if (s.length() % 2 == 1)
3.          return false;
4.      for (int i = 0; i < s.length() - 1; i = i + 2)
5.          if (s.charAt(i) != s.charAt(i + 1))
6.              return false;
7.      return true;
8.  }
```

A. isBRStr("abababababa")

B. isBRStr("abababbabab")

C. isBRStr("aabbccdd")

D. isBRStr("aabb")

5. Which option's return value is true to the code fragment below?

```
1.  public static boolean checkStrTwoSide(String str){
2.      if (str.length() % 2 == 1)
3.          return false;
4.      for (int i = 0; i < str.length() / 2; i++)
5.          if (str.charAt(i) != str.charAt(str.length() - 1 - i))
6.              return false;
7.      return true;
8.  }
```

A. checkStrTwoSide("hello")

B. checkStrTwoSide("AHA")

C. checkStrTwoSide("ABBA")

D. checkStrTwoSide("Hi")

6. What is the output of the following program fragment?

```
1.  public class DealStr {
2.      public static String addStr(String str){
3.          StringBuilder ans = new StringBuilder();
4.          for (int i = 0; i < str.length(); i++){
5.              ans.append(str.charAt(i));
6.              ans.append(str.charAt(str.length() - 1 - i));
7.          }
8.
9.          return String.valueOf(ans);
10.     }
11.
12.     public static void main(String[] args) {
13.         System.out.println(addStr("aaabbb"));
14.     }
15. }
```

A. aaaaaabbbbbb

B. ababab

C. abababbababa

D. bababa

Section C Coding (56 marks)

1. (15 mark) Write a code fragment which uses nested for-loops to output the following pattern.

```
#
##
###
####
#####
```

2. (15 marks) Largest Prime

Write a method named **getLargestPrime** with one parameter of type **int** named number.

If the **number** is negative or does not have any prime numbers, the method should **return -1 to indicate an invalid value**.

The method should **calculate the largest prime factor of a given number** and return it.

EXAMPLE INPUT/OUTPUT:

- `getLargestPrime (21);` should **return 7** since 7 is the largest prime (3 * 7 = 21)
- `getLargestPrime (217);` should **return 31** since 31 is the largest prime (7 * 31 = 217)
- `getLargestPrime (0);` should **return -1** since 0 does not have any prime numbers
- `getLargestPrime (45);` should **return 5** since 5 is the largest prime (3 * 3 * 5 = 45)
- `getLargestPrime (-1);` should **return -1** since the parameter is negative

HINT: Since the numbers 0 and 1 are not considered prime numbers, they cannot contain prime numbers.

NOTE: The method **getLargestPrime** should be defined as **public static** like we have been doing so far in the course.

NOTE: Do not add a **main** method to the solution code.

3. (26 Marks) Interface

Create a simple interface that allows an object to be saved to some sort of storage medium.

Implement the following:

1. ISaveable (interface)

- It has two methods:
  - write(), takes no arguments and returns a List containing objects of type String.
  - read(), takes a List of type String and doesn't return anything.

2. Player (class)

- It has four fields. Two Strings called name and weapon. Two ints called hitPoints and strength.

- A constructor that accepts a String (name) and two ints (hitPoints and strength). It initialises name, hitPoints and strength with the newly passed in values. It initialises weapon with the default weapon "Sword".

- And eleven methods:
  - Getters and setters for all four fields.
  - write(), same as interface. Return a List of the fields in the order they appear in toString().
  - read(), same as interface. Store the values in the List, in the order they appear in toString(). Make sure the List is not null and the size() is greater than 0 before storing the values.
  - toString(), Players *overriding* toString() method. It takes no arguments and returns a String in the following format:

```
Player{name='Tim', hitPoints=10, strength=15, weapon='Sword'}
```

3. Monster (class)

- It has three fields. One String called name and Two ints called hitPoints and strength.

- A constructor that accepts a String (name) and two ints (hitPoints and strength). It initialises name, hitPoints and strength with the newly passed in values.

- And six methods:
  - Only getters for the three fields.
  - write(), same as interface. Return a List of the fields in the order they appear in toString().

- read(), same as interface. Store the values in the List, in the order they appear in toString(). Make sure the List is not null and the size() is greater than 0 before storing the values.

- toString(), Monsters *overriding* toString() method. It takes no arguments and returns a String in the following format:

```
Monster{name='Werewolf', hitPoints=20, strength=40}
```

TIP: Player and Monster need to implement ISaveable.

TIP: Be extremely careful with the spelling of the names of the fields, constructors and methods.

TIP: Be extremely careful about spaces and spelling in the returned String from the toString() method.

NOTE: All fields are **private**.

NOTE: Both constructors are **public**.

NOTE: All methods are **public**.

NOTE: There are no **static** members.

NOTE: If you get an error from the *Evaluate* class, it's most likely the constructor. Check if you've added a constructor or if the constructor has the right arguments.

}

----------End of Paper----------