

CITY UNIVERSITY OF HONG KONG

Course code & title : CS2312 Problem Solving and Programming

Session : Semester A 2019-20

Time allowed : Two (2) Hours

This paper has 6 pages (including this page).

Note:

1. There are 5 questions in this paper.
2. Answer ALL questions in the answer paper provided.

Remarks:

- Apply best practices and skills of programming when you answer the questions.
 - Marks may be deducted for redundant or unnecessary code.
 - You do NOT need to handle input errors or exceptions unless explicitly mentioned in the question.
 - Do not change or delete any given code.
-

*This is a **closed-book** examination.*

*No materials or aids are allowed during the whole examination.
If any unauthorized materials or aids are found on a candidate during the examination, the candidate will be subject to disciplinary action.*

THIS PAPER IS
NOT TO BE
TAKEN AWAY

Question 1 (20 marks)

(a) What is the output of the following program?

(6 marks)

Show your work by drawing the contents and changes in all created objects.

(4 marks)

<pre>public class Value { private int i; public Value(int i) { this.i = i; } public void advance() { i = i * 2; } public void tellValue() { System.out.println(i); } }</pre>	<pre>public class MultiValue extends Value { private Value child = null; public MultiValue(int i, Value child) { super(i); this.child=child; } @Override public void advance() { super.advance(); child.advance(); } @Override public void tellValue() { super.tellValue(); child.tellValue(); } }</pre>
<pre>public class Main { public static void main(String args[]) { Value v1 = new Value(100); //Statement S1 Value v2 = new MultiValue(300, v1); //Statement S2 Value v3 = new MultiValue(500, v2); //Statement S3 v1.advance(); //Statement S4 v2.advance(); //Statement S5 v3.advance(); //Statement S6 v3.tellValue(); //Statement S7 } }</pre>	

(b) Using any statement from the program in part (a), explain the term **Polymorphism**.

(4 marks)

(c) Briefly explain **up-casting** and **down-casting**.

Is there any statement doing up-casting or down-casting in the main method in part (a)? If yes, which kind of casting? In which statement(s)?

Using the classes Value and MultiValue, write an example where **explicit casting** is needed.

(6 marks)

Question 2 (15 marks)

- (a) Add required classes / interfaces to the program below such that the whole program is free of error and produces output as given in the comments in main(). Note that you should give the solution in **minimal code**.

(11 marks)

```
public class Counter {
    private int count=0;
    private Controller k;
    public Counter(Controller k) {this.k=k;}
    public void tick() {k.tick(this);}
    public void increment(int incr) {
        count+=incr;
        System.out.printf("Increment = %d,  result = %d\n",
            incr, count);
    }
}

public class Main {
    public static void main(String[] args) {
        Counter counter1 = new Counter(new Simple());
        counter1.tick(); //Output: Increment = 1,  result = 1
        counter1.tick(); //Output: Increment = 1,  result = 2
        counter1.tick(); //Output: Increment = 1,  result = 3
        counter1.tick(); //Output: Increment = 1,  result = 4
        counter1.tick(); //Output: Increment = 1,  result = 5

        Counter counter2 = new Counter(new Advanced());
        counter2.tick(); //Output: Increment = 1,  result = 1
        counter2.tick(); //Output: Increment = 2,  result = 3
        counter2.tick(); //Output: Increment = 3,  result = 6
        counter2.tick(); //Output: Increment = 4,  result = 10
        counter2.tick(); //Output: Increment = 5,  result = 15
    }
}
```

- (b) State 2 reasons for using an abstract class instead of an interface in Java programming.

(4 marks)

Question 3. (20 marks)

We are to rate the dishes cooked by different persons (i.e., the *cooks*) for various *recipes*. We will list all scores obtained by each cook, and find out the best cook of each recipe. The scores are non-zero positive integers. A higher score means that the cook is better.

The main program below creates objects for cooks, recipes, and dishes. Also, it gives scores to the dishes and reports the result. The expected outputs are given as underlined contents in the inline comments in **main()**.

```
public static void main(String[] args) {
    Cook c1 = new Cook("Candy");
    Cook c2 = new Cook("Cara");
    Recipe r1 = new Recipe("Grilled Fish");
    Recipe r2 = new Recipe("Baked beancurd");
    Recipe r3 = new Recipe("Minced cabbage");
    Dish d1 = c1.cookDish(r1);
    Dish d2 = c1.cookDish(r2);
    Dish d3 = c2.cookDish(r2);

    d1.obtainScore(5);
    d2.obtainScore(4);
    d3.obtainScore(6);

    c1.report(); //Output: I got 2 scores: 5 4
    c2.report(); //Output: I got 1 scores: 6

    r1.report(); //Output: Grilled Fish: The best cook is Candy
    r2.report(); //Output: Baked beancurd: The best cook is Cara
    r3.report(); //Output: Minced cabbage: [No data]
}
```

For simplicity, assume the scores are distinct.

Your task: Finish the classes **Cook**, **Recipe**, and **Dish**.

Guideline:

- A dish contains references to its recipe and its cook.
- To store the scores obtained by each cook, we use an `ArrayList` of `Integers` in each cook object. Whenever a dish obtains a score, the cook is told to store it.
- To keep track of the best cook of each recipe, the recipe contains a reference to the best cook, and correspondingly the best score number. These two fields are checked and updated as needed whenever any dish of the recipe obtains a score.

Question 4 (20 marks)

A barber runs a hair salon alone. Each day he provides service to at most 6 customers only, starting from 11:00, one hour is allocated to each customer, assume no lunch break. The program below creates objects for the barber's customers who call to come on a day. There are two types of customers:

- Customer, the basic type, tells the wanted service time.
- CustomerEarly, another type of customer, just wants the earliest time possible.

- (a) The main method and the Customer class are given below. Study the code, the comments and program output. Also study the given code in the Barber and CustomerEarly classes on the answer paper. Add all missing code in the Barber and CustomerEarly classes. You should implement the Barber class using the **Singleton Pattern**.

(17 marks)

For simplicity, we assume that all requests are successfully scheduled.

```
public static void main(String[] args) {
    Barber b = Barber.getInstance();

    (new CustomerEarly("Mr POON")).call(b);
    (new CustomerEarly("Ms TANG")).call(b);
    (new Customer("Ms WONG")).call(b, 1500);
    (new CustomerEarly("Ms CHAN")).call(b);

    b.list(); //output the customer list
}
```

Program output:

```
Mr POON (1100)
Ms TANG (1200)
Ms CHAN (1300)
Ms WONG (1500)
```

```
public class Customer {
    private String name;
    private int serviceTime;

    public Customer(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return this.name + " (" + this.serviceTime + ")";
    }

    public void call(Barber b, int t) {
        serviceTime = t;
        b.markTime(this, t);
    }

    protected void setServiceTime(int t) {
        serviceTime = t;
    }
}
```

- (b) Explain how your code for the **Barber** class achieves the purpose of the **Singleton Pattern**.

(3 marks)

Question 5 (25 marks)

Given the main() method for testing and the Customer class, complete the remaining classes and interfaces for the program according to the following instructions. The program should run as shown in the output below.

- The program creates objects for Products and Services, which are provided by Shops and ordered by Customers.
- Products and Services all belong to Tradables.

- All Tradables have ID and price.

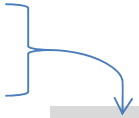
In addition, 10% discount is given to products which are over 50 dollars, and 20% discount is given to services which are over 100 dollars.

Also, the profits for products are 10%, and the profits of services are 30%.

- To customers, the Tradables are known as objects of the type IConsumable. Customers should not know about the profits. Therefore the Customer class does not depend on the class for Tradables directly.

The main() method:

```
public class Main {  
    public static void main(String [] args)  
    {  
        Product p1 = new Product("P001", 100); //id and price  
        Product p2 = new Product("P002", 200);  
        Product p3 = new Product("P003", 300);  
  
        Service s1 = new Service("S001", 150); //id and price  
        Service s2 = new Service("S002", 250);  
        Service s3 = new Service("S003", 350);  
  
        Customer c1 = new Customer("Helena"); //customer name  
        Customer c2 = new Customer("Mary");  
        Customer c3 = new Customer("Paul");  
  
        Shop h1 = new Shop("Shop A", new Tradable[] {p1,s1}); //Shop name, products and services  
        Shop h2 = new Shop("Shop B", new Tradable[] {p1,p2,p3});  
        Shop h3 = new Shop("Shop C", new Tradable[] {s1,s2,s3});  
  
        c1.order(s1, h1);  
        c2.order(p1, h2);  
        c3.order(p2, h3);  
    }  
}
```



Output:

Helena orders S001 from Shop A, price is 150, discount percentage is 20
Traded: S001, Earning: 45

Mary orders P001 from Shop B, price is 100, discount percentage is 10
Traded: P001, Earning: 10

Paul orders P002 from Shop C, price is 200, discount percentage is 10
Not provided!

The Customer class:

```
class Customer {  
    private String name;  
    public Customer(String name) {this.name=name;}  
    public void order(IConsumable cons, Shop s) {  
        System.out.println(name + " orders " + cons + " from " + s +  
            ", price is " + cons.getPrice() +  
            ", discount percentage is " + cons.getDiscountPercent());  
        s.processOrder(this, cons);  
    }  
}
```