

EXAMINATIONS – 2016

TRIMESTER 1

COMP 102
INTRODUCTION TO
COMPUTER PROGRAM
DESIGN

Time Allowed: TWO HOURS

CLOSED BOOK

Permitted materials: Silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Printed foreign language–English dictionaries are permitted.

No other material is permitted.

Instructions: Attempt ALL Questions.

The exam will be marked out of 120 marks.

Brief Java Documentation will be provided with the exam script

Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.

There are spare pages for your working and your answers in this exam, but you may ask for additional paper if you need it.

Questions

	Marks
1. Understanding Java	[32]
2. Files and ArrayLists	[22]
3. Defining Interfaces and Classes	[20]
4. ArrayLists of Objects	[26]
5. Arrays	[20]

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. Understanding Java**(32 marks)****(a) (6 marks) Loops and calling methods.**

What will the following printNumbers method print out?

```
public void printNumbers(){
    int m = 10;
    this.doPrint(m);
    UI.println (m);
}
public void doPrint(int x){
    while (x < 100){
        x = x * 2;
        UI.println ("x: " + x);
    }
    UI.println ("finally "+ x);
}
```

Write your answers here

(b) (6 marks) Calling methods on objects.

Consider the following Score class:

```
class Score {  
    private int num;  
  
    public Score(int x ){  
        this.num = x;  
    }  
  
    public void testA(){  
        if ((this.num <3) || (this.num >10)){  
            Ul.println ("Yes");  
        }  
        else {  
            Ul.println ("No");  
        }  
    }  
  
    public boolean testB(int y){  
        return (this.num < y);  
    }  
}
```

What will the following useObjects method print out?

```
public void useObjects(){  
    Score sc1 = new Score(15);  
    Score sc2 = new Score(10);  
    sc1.testA ();  
    Ul.println (sc1.testB(12));  
  
    if (sc2.testB(13))  
        Ul.println ("good");  
    else  
        Ul.println ("better");  
}
```

Write your answers here

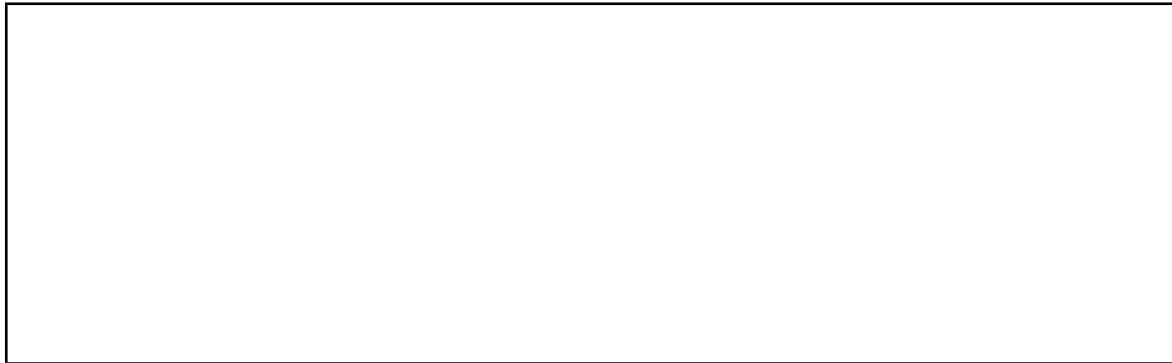
(c) (6 marks) Arrays.

What will the following arrayTest method print out?

```
public void arrayTest(){
    String [ ] cities = new String [ ]{"Syd", "Auck", "Mel", "Wlg", "Cairns"};

    int n = 3;
    cities [n] = cities [n+1];

    for( int i=0; i < cities .length; i++){
        if ( cities [i].length()>3){
            UI. println ( cities [i ]);
        }
    }
}
```



(d) (6 marks) Using 2D Arrays.

What will the following array2DTest method print out?

```
public void array2DTest(){
    int [ ][ ] nums = new int [ ][ ]{{ 1, 3, 5, 7},
                                       {11, 14, 17, 20},
                                       {21, 25, 29, 33}};

    Ul.println ( nums[0][3] );
    Ul.println ( nums[2][0] );

    Ul.println ( "-----");
    nums[1][1] = nums[2][2];

    for ( int k = 0; k < 3; k++){
        Ul.println ( nums[k][k] );
    }
}
```

Write your answers here

(e) (8 marks) Using ArrayLists.

What will the following arrayListTest method print out?

```
public void arrayListTest(){
    ArrayList <Double> nums = new ArrayList<Double>();
    nums.add(2.4);
    nums.add(5.2);
    nums.add(7.5);
    nums.add(10.0);
    UI.println (nums.size());
    UI.println (nums.get(0));
    UI.println (nums.indexOf(10.0));
    UI.println ("-----");
    nums.add(1, 20.3);
    for(double n : nums){
        UI.println (n);
    }
    UI.println ("-----");
    nums.set(2, 11.1);
    nums.remove(3);
    for(double n : nums){
        UI.println (n);
    }
}
```

Write your answers here

Question 2. Files and ArrayLists**(22 marks)**

This question concerns a class with methods for processing files that contain a mixture of numbers and text. The text can be anywhere in the file and can be of any length. The following is an example of a file that this class can process:

```
5 9 some notes 1.5 2.6 here 10 20 8 8
and 30 20 60 there 20.3 50 every where 70.1
12 40 in any length 40 50 15 70 15.4 19
```

Suppose the class has a `nums` field defined as follows

```
private ArrayList<Double> nums = new ArrayList<Double>();
```

(a) (10 marks) Complete the following `readToArrayList` method that adds every number in a file into the `ArrayList` in the `nums` field. The parameter of the `readToArrayList` method is the name of the file to read.

```
public void readToArrayList(String filename){

    try{

    } catch(IOException e){UI.println("error: " + e);}

}
```


(b) (12 marks) Complete the following `writeToFile` method that writes the numbers in the `nums` field into a new file. The name of the new file is specified in the parameter.

The first line in the new file should contain just one integer, which specifies how many numbers are in the file.

The rest of the file should contain all the numbers in the `ArrayList`. For full marks there should only be 5 numbers on each line.

For example, using the example file on the facing page, the new file should contain

```
22
5.0 9.0 1.5 2.6 10.0
20.0 8.0 8.0 30.0 20.0
60.0 20.3 50.0 70.1 12.0
40.0 40.0 50.0 15.0 70.0
15.4 19.0
```

Hint: write the numbers with `print`, and after every fifth number, do a `println()`;

```
public void writeToFile(String filename){
    try{

    } catch(Exception e){UI.println("file error");}
}
```


(b) (12 marks) Complete the following `AdultMember` class.

This class should implement the `Member` interface and work well with the code given on the facing page.

You will need to define the appropriate fields, a constructor and all the necessary methods.

- The constructor should have parameters for the member's name and number.
- The `printInvoice` method can be very simple, e.g. prints a one line message "pay \$20 fortnightly" for all adult members.
- The `topUpAccount` method should add an amount to the member's account balance which should be stored in a field.
- The `toString` method should include the name, number, and account balance.
- The other methods are very straight forward.

```
class AdultMember
```

```
}
```

Question 4. ArrayLists of Objects**(26 marks)**

This question concerns a program for managing the loyalty programme for customers of a company. The program has two classes: a `Customer` class and a `LoyaltyProgramme` class.

The `Customer` class below defines `Customer` objects, which store the customer names and the points they have earned.

```
class Customer {  
    private String name;  
    private int points = 0;  
  
    public Customer(String s) {  
        this.name = s;  
        this.points = 0;  
    }  
  
    public void updatePoints(int n) {  
        this.points = this.points + n;  
    }  
  
    public int getPoints() {  
        return this.points;  
    }  
    public String getName() {  
        return this.name;  
    }  
  
    public String toString(){  
        return (this.name + " : " + this.points + " points ");  
    }  
}
```

(Question 4 continued on next page)

(Question 4 continued)

The `LoyaltyProgramme` class defines a field to store the list of `Customer` objects:

```
private ArrayList<Customer> customers = new ArrayList<Customer>();
```

You may assume that `customers` will never contain any null values.

The `LoyaltyProgramme` class also defines several methods.

(a) (12 marks) Complete the following `printVouchers` method that prints a voucher for each customer in the loyalty programme who has earned more than 200 points.

The amount of the voucher depends on the points: \$10 is awarded for every 200 points. For example, if a customer has earned 600 points, the voucher should be \$30.

Print the name and the amount for each voucher. For full marks, the vouchers should always be for a multiple of \$10 — with 430 points, the voucher should be just \$20.

```
public void printVouchers(){
```

```
}
```

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 4 continued)

(b) (14 marks) Complete the following `earnPoints` method that calculates and updates the points for a purchase by a customer. The first parameter is the customer name. The second parameter is the amount that the customer has spent. The customers earn 1 point for each \$20 they spend.

If the customer is not found in the database, `earnPoints` should print a "new customer" message and automatically add this customer to the database and update the customer's points.

After the points are updated, it should print the details of the customer.

```
public void earnPoints(String name, double amount){
```

```
}
```

Question 5. Arrays**(20 marks)**

Consider the following program with a field called `hand` which contains an array of up to 10 cards.

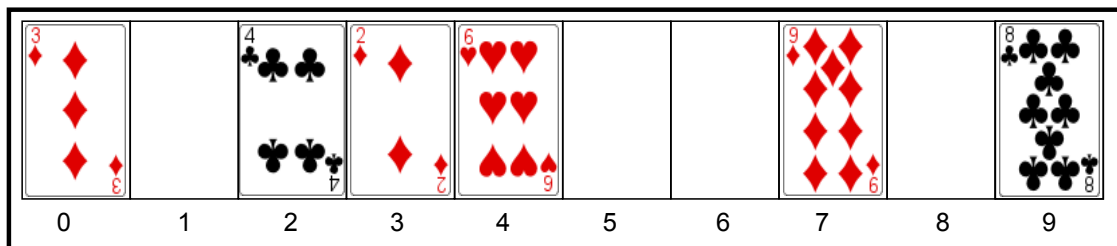
The `startHand` method adds some cards to the array.

You can assume the `Card` class has a constructor with two parameters specifying the card suit (diamonds, hearts, spades, clubs) and rank (1-13).

```
private Card[] hand = new Card[10];

public void startHand() {
    this.hand[0] = new Card("diamonds", 3);
    this.hand[2] = new Card("clubs", 4);
    this.hand[3] = new Card("diamonds", 2);
    this.hand[4] = new Card("hearts", 6);
    this.hand[7] = new Card("diamonds", 9);
    this.hand[9] = new Card("clubs", 8);
}
```

After calling `startHand`, the array would look like this:



(a) (5 marks) What does the following testPrint method print out? Very briefly describe what it does in one sentence.

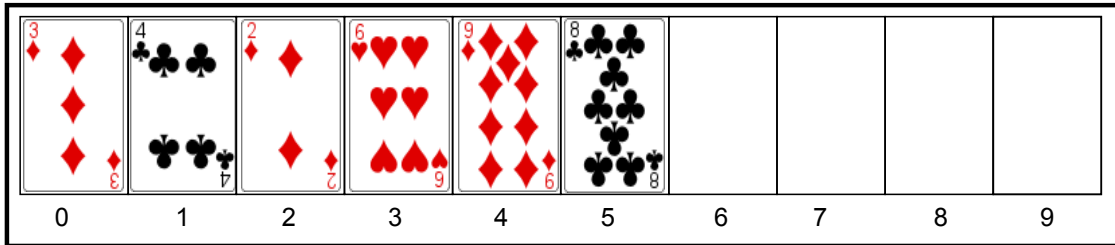
```
public void testPrint () {  
    int count = 0;  
  
    for (int i = 0; i < this.hand.length; i++){  
        if (this.hand[i] == null){  
            count ++;  
        }  
    }  
    UI.println (count);  
}
```

Output:

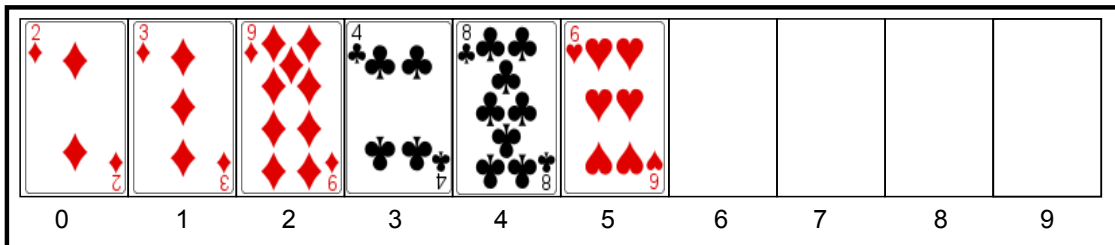
Description:

(b) (15 marks) Complete the following compact method that moves all the Cards as far to the left as possible, so that the Cards fill the first part of the array and any nulls are in the rest of the array.

For example, after running this method on the example array shown before, the array should now look like this:



For full marks, you should put all cards with the same suit together, and sort the card in each suit in increasing order of the ranks. For example, it would be better if your array looks like this:



You may assume the Card class has `getSuit` and `getRank` methods that return the suit and the rank of a Card.

```
public void compact(){
```

```
}
```
