

Project 1: Java Homework

Hao Wu
5140219226

May 16, 2017

Instructor: Ling Gong

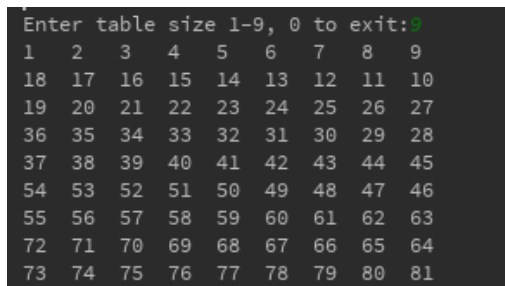
Project 1-1 <Snake-Like Increment Table>

1.1 Object:

This program will create small, snake-like increment tables, up to 9x9 in size. It starts from 1, and the value keep increasing by one. For the first row, it increase from left to right; for the second row, it increase from right to left; the third row, from left to right, ... , etc. Just like a snake.

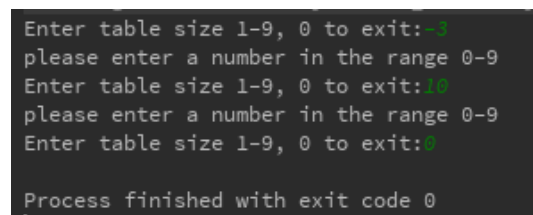
1.2 Screen Cut:

When we input the table size between 1-9, we can obtain the snake-like increment table as figure1.1. And once the input number is below 1 or over 9, the console will return “please enter a number in the range 0-9”, and only if the input number is 0, the process will finish, as figure 1.2 shows.



```
Enter table size 1-9, 0 to exit:
1  2  3  4  5  6  7  8  9
18 17 16 15 14 13 12 11 10
19 20 21 22 23 24 25 26 27
36 35 34 33 32 31 30 29 28
37 38 39 40 41 42 43 44 45
54 53 52 51 50 49 48 47 46
55 56 57 58 59 60 61 62 63
72 71 70 69 68 67 66 65 64
73 74 75 76 77 78 79 80 81
```

Fig1.1 Snake-like increment table



```
Enter table size 1-9, 0 to exit:-1
please enter a number in the range 0-9
Enter table size 1-9, 0 to exit:10
please enter a number in the range 0-9
Enter table size 1-9, 0 to exit:0
Process finished with exit code 0
```

Fig1.2 Error processing

Project 1-2 <Date Processing>

1.3 Object:

This problem is much more flexible, any reasonable implementation is acceptable. Please implement a class called MyDate which can represent a date. The class should have at least three int fields: year, month, day, and several methods to handle them. Please make sure the values set to these fields are all valid. Pay attention to the special cases: some month can only 28/29 days, and some month can have 31 days.

1.4 Screen Shot

Fig1.3 & Fig1.4 show the basic function of this program, comparing two date to decide which is early and calculating the gap between two days. Fig1.5 presents the error conducting, if you enter a wrong number, the console will return “The data should be like the example 9/15/2014”, and only if you enter 0, the program will finish.

```

1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:1
Please enter the first date DateA! The date should be like the example 9/15/2004:1/25/2015
Please enter the second date DateB! The date should be like the example 9/15/2004:1/26/2014
Date1 is later than date2!

```

Fig1.3 Date comparison

```

1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:2
Please enter the start date! The date should be like the example 9/15/2004:1/25/2015
Please enter the end date that is later than the start! The date should be like the example 9/15/2004:1/26/2020
1845

```

Fig1.4 Date gap calculating

```

1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:1
Please enter the first date DateA! The date should be like the example 9/15/2004:2/31/2015
The date should be like the example 9/15/2004:1/31/2015
Please enter the second date DateB! The date should be like the example 9/15/2004:2/25/2015
Date1 is NOT later than date2!
1 stands for judging if date A is later than date B
2 is calculating the days between date A and B
0 means to exit
Choose a function:0
Process finished with exit code 0

```

Fig1.5 Error processing

Project 2 <Bank Account>

2.1 Object:

Suppose you are opening your own tiny bank, and want to manage the bank accounts in your bank. In your bank, there are only two kinds of account: checking and saving. Both of them should have an account number, and should be able to perform deposit, withdraw and check balance operations. People can open a new account with some money deposited, or no money at all. Checking account and saving account perform differently in some aspects, and there are some requirement to meet about the bank accounts.

2.2 Screen Shot:

We can deposit, withdraw, check balance for each checking accounts, saving accounts by choosing the corresponding function number. And when we choose back, the benefits and poundage for each account per month will be summed. For each account, you can't type in any number not belonged to 0~3, and only if you enter 0, the program will be ended.

```
my checking number is 1, my balance is 38.0
my checking number is 2, my balance is 38.0
my checking number is 3, my balance is 38.0
my checking number is 4, my balance is 38.0
my checking number is 5, my balance is 38.0
my saving number is 6, my balance is 94.5
my saving number is 7, my balance is 94.5
my saving number is 8, my balance is 94.5
my saving number is 9, my balance is 94.5
my saving number is 10, my balance is 94.5
Welcome to the bank system in the next month! Please choose what you want to do!
1: choose a bank account
0: exit
Your choice:1
Please input the bank account:0
What's next?
1: Deposit
2: Withdraw
3: Balance
0: back
Your choice:1
How much do you want to deposit:100
What's next?
1: Deposit
2: Withdraw
3: Balance
0: back
Your choice:3
my saving number is 6, my balance is 194.5
What's next?
1: Deposit
2: Withdraw
3: Balance
0: back
Your choice:2
How much do you want to withdraw:50
What's next?
1: Deposit
2: Withdraw
3: Balance
```

Fig2.1 Bank algorithm function

```
Please input the bank account:1
What's next?
1: Deposit
2: Withdraw
3: Balance
0: back
Your choice:4
Please input a number between 0 and 2!
```

Fig2.2 Error processing

Project 3 <Worker System>

3.1 Object:

This homework revisits the idea of superclass/subclass, polymorphism. And study how to use Arrays.sort(). Remember you need to implement Comparable or Comparator to sort your object array.

Write a superclass `Worker` and subclasses `HourlyWorker` and `SalariedWorker`. Every worker has a name, a salary rate, and the total salary he got so far from the beginning of his employment. Write a method `computePay(int hours)` that computes the weekly pay for every worker. An hourly worker get paid the the hourly wage for the actual number of hours worked, of hours is at most 40. If the hourly worker worked more than 40 hours, the excess is paid at double rate. The salaried worker gets paid the hourly wage for 40 hours, no matter what the actual number of hours is. The total salary is just the accumulated value of all the paid salary for the worker. Beside the `computePay()` method, you can add more methods if you like or think necessary.

After creating an array of `Worker`, I want to sort the array and print out the sorted result. By default, the workers are sorted by their name. You should also be able to sort it by their salary rate, or by the total salary they've got. This means if I use `Arrays.sort(arrayname)` directly, I'm sorting by name. Otherwise I sort by using different comparators.

When print out the information about the workers using `System.out.println(worker)`, I wish to be able to see the worker's type too, i.e., whether he is `HourlyWorker` or `SalariedWorker`, and with his name, salary rate, and total salary.

3.2 Screen Shot:

We can choose how many workers work this week and the type of each worker, and for hourlyWorkers we can decide his work time this week. After this week, we can count their salary like Fig3.1 gives, and sort them by different orders. Only if you enter 0, the program will exit.

```
How many workers?: 2
Please input the type of the worker (1-HourlyWorker; 2-SalariedWorker): 1
Please input the name of the worker: Wu
Please input the salary rate of the worker: 2
Please input the type of the worker (1-HourlyWorker; 2-SalariedWorker): 2
Please input the name of the worker: Da
Please input the salary rate of the worker: 3

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 1
Please input how many hours does the worker Wu work in this week: 60
Wu : 160.0

Da : 120.0

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 2
Type      Name      Salary Rate Total Salary
HourlyWorker  Wu      2.0      160.0
SalariedWorker Da      3.0      120.0
```

Fig3.1 Salary count function

```

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 4

1:Sort by name
2:Sort by salary rate
3:Sort by total salary
Your choice: 1
Type      Name      Salary Rate Total Salary
SalariedWorker Da  3.0      120.0
HourlyWorker  Wu  2.0      160.0

1:Get the weekly salary
2:Get the worker's information
3:Find a worker
4:Sort in order
0:exit
Your choice: 4

1:Sort by name
2:Sort by salary rate
3:Sort by total salary
Your choice: 3
Type      Name      Salary Rate Total Salary
SalariedWorker Da  3.0      120.0
HourlyWorker  Wu  2.0      160.0

```

Fig3.2 Order Function**Project 4.1 < Socket NetWork>****4.1 Object:**

Write a pair of programs that will communicate over a socket, one called AddClient and one called AddServer. AddServer takes one command-line parameter, the port it should listen on. AddClient takes three command-line parameters: the address of the server, the port of the server and a string to send to the server. Each run of AddClient will send one string to the server. AddServer repeatedly listens for Strings from clients. When it hears one, it attempts to convert it to an int and add it to its counter (initially 0). If the conversion fails, it just waits for the next connection. If the received String equals "exit", AddServer prints the value of counter to the screen and exits. Both programs can exit on errors (cleanly, with a message), but AddServer must tolerate NumberFormatExceptions (use try-catch block to handle it).

4.2 Screen Shot:

We can see the client and server are binding by the port 2000 in localhost, and now we can send number from client to server, and if the sent data is character, the server will return "Error" compared with "Roger" if succeed. If we enter "exit", the server will give the sum of pre entered number, as Fig4.2 shows "Counter =

23". And only if we send "exit" to server can we cut off the connection between client and server.

```
wu@wu: ~/Desktop/javaProject/JavaHomework/project4_AddClient/src
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ javac
AddClient.java
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ java A
ddClient localhost 2000 8
Error:java.net.ConnectException: Connection refused (Connection refused)
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ java A
ddClient localhost 2000 8
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ java A
ddClient localhost 2000 5
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ java A
ddClient localhost 2000 10
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ java A
ddClient localhost 2000 ee
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ java A
ddClient localhost 2000 exit
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddClient/src$ |
```

Fig4.1 Add client operation

```
wu@wu: ~/Desktop/javaProject/JavaHomework/project4_AddServer/src
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddServer/src$ javac A
.java
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddServer/src$ java Ad
2000
Roger!
Roger!
Roger!
Error!
counter = 23
wu@wu:~/Desktop/javaProject/JavaHomework/project4_AddServer/src$ |
```

Fig4.2 Add server responding

Project 4.2 < I/O and Network Downloading>

4.3 Object:

Create a class WGet that takes a URL and a filename on the command line, and attempts to retrieve the data at that URL into the specified file. For example, "java WGet remoteWebPage.html myfile.html", then you are retrieving the file at CS website and store it into your own file my f ile.html. It should fail if the filename you are trying to write to is already existing, and fail with an informative message if something goes wrong (i.e. don't just dump a stack trace). You can assume that the data is binary, i.e. don't worry about whether to use a Reader, just use an InputStream. You can also assume the file is text only. Just notify it in your code.

4.4 Screen Shot:

The function WGet can download the remote html source to local disk as myfile.html, and anytime we open the myfile.html locally, we can see the pure rendered web page by browser ignoring css framework or javascript. As shown in the figure4.3, I download the <http://www.bilibili.com/> to bilibili.html locally, and finally be able to observe it by browser.

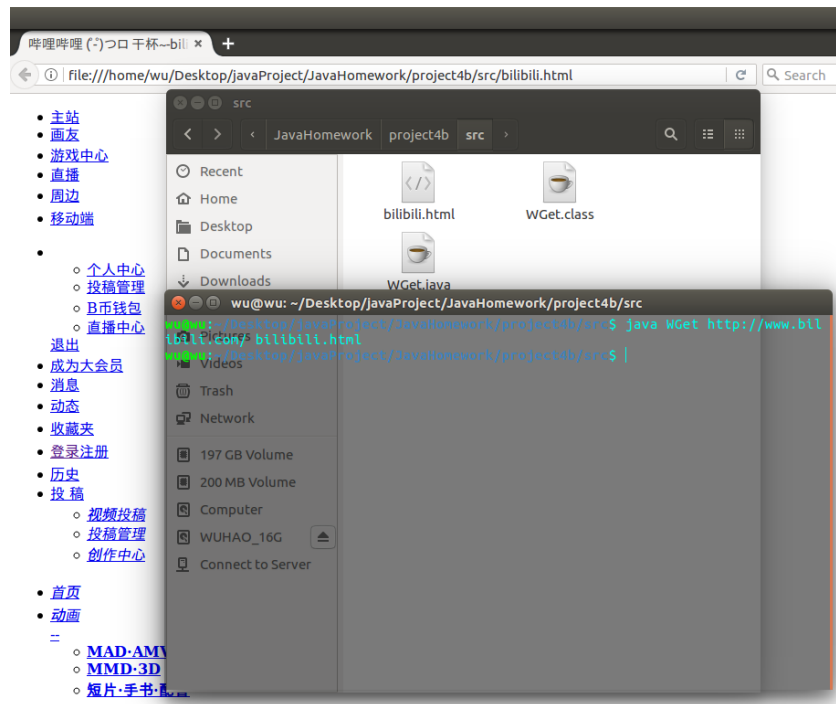


Fig4.3 Get the remote web source to local disk

Project 5.1 < Multi-Thread Programming >

5.1 Object:

Create a class named Countdown that uses an explicit Thread (don't use the Timer class for this homework) to write a countdown to the console, one number per second. You can either extend Thread or implement Runnable. Your solution is free to implement other classes, but main() should be in Countdown.

5.2 Screen Shot:

Create a class named Countdown that uses an explicit Thread (don't use the Timer class for this homework) to write a countdown to the console, one number per second. You can either extend Thread or implement Runnable. Our solution is free to implement other classes, but main() should be in Countdown. The program will not actually exit until the user hits the enter key. If the user hits the enter key before the countdown has finished, the program should print interrupted and stop immediately.

```
wu@wu:~/Desktop/javaProject/JavaHomework/project5/src$ java project5
10
9
user hits the Enter key
interrupted
Finished
```

Fig5.1 Interrupt by keyboard


```
wu@wu: ~/Desktop/JavaProject/JavaHomework/project5/src$ java project5
10
9
8
7
6
5
4
3
2
1
0
user hits the Enter key
Finished
```

Fig5.2 Error processing

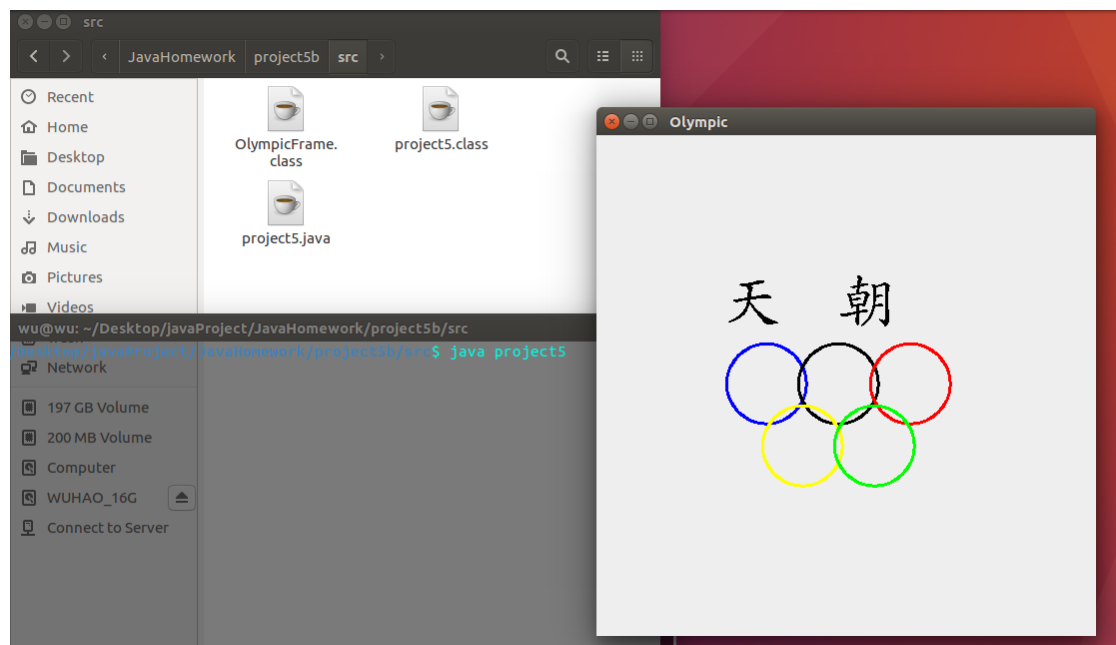
Project 5.2 < GUI Programming >

5.3 Object:

Write a program that displays the Olympic rings. Make the rings colored in the Olympic colors. If you like to make it fancier, you can add "USA" on top the rings (not required). You are not required to draw exactly the same as in the pictures. Anything looks similar is fine. And the color in the intersection part of the rings is not strictly required.

5.4 Screen Shot:

As shown in the figure5.1, we can observe that in the set frame, five circles with different color and title are presented.

**Fig5.3 GUI interface**