DEBRE BERHAN UNIVERSITY

INSTITUTION of TECHNOLOGY

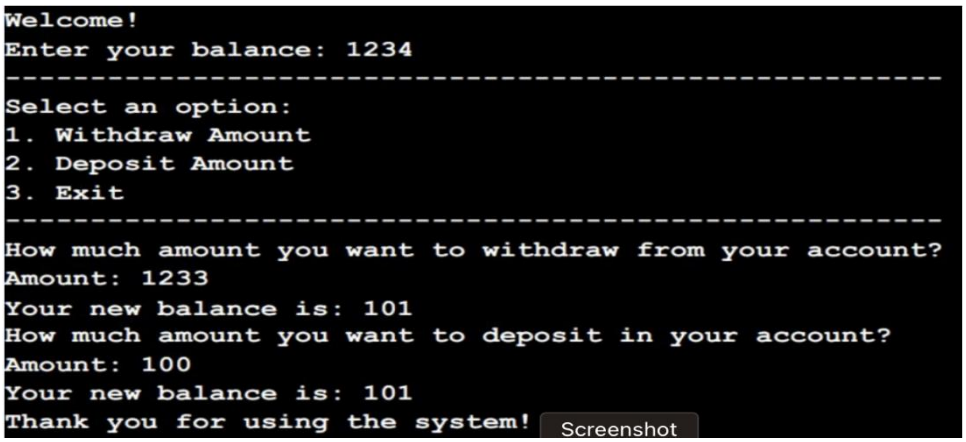
COLLAGE of COMPUTING

DEPARTMENT of SOFTWARE ENGINEERING

OBJECT-ORIENTED SOFTWARE ENGINEERING

No	Name	Id
1	1.Yewoynehareg Mulugeta	0035/13
2	2.Eyerusalem Fikadu	0395/13
3	3.Firdiwek Sisay	1510/13
4	4.Hassen Muhamed	0280/13
5	5.Eyob Chiksa	0489/13
6	6.Fasiledes Shimelis	0105/13
7	7.Fasika	4100/13
8	8.Khader yare	3689/13

1. Write a program that takes balance of a user's account as input. It should then ask the user how much amount he wants to withdraw from his account. The program should take this amount as input and deduct from the balance. Similarly it should ask the user how much amount he wants to deposit in his account. It should take this amount as input and add to the balance. The program shall display the new balance after amount has been withdrawn and deposited. Note: Your program should have a check on balance and amount being withdrawn. Amounts greater than balance cannot be withdrawn i.e. balance cannot be negative.



Solution

```
import java.util.Scanner;

class checker {
void check(double Amount) {
if (Amount <= 0) {
System.out.println("invalid Amount!");
System.out.println("please enter amount that is grtaer than
0");
}
}
}
class Account {
public static void menu() {
System.out.println("-----------------------------------------
--------------------------------");
System.out.println("Select an option:");
```

```
System.out.println("1. Withdraw Amount");
System.out.println("2. Deposit Amount");
System.out.println("3. Balance");
System.out.println("4. Exit");
System.out.println("----------------------------------------
------------------------------");
}
public static void main(String[] args) {
checker check = new checker();
System.out.println("Wellcome!");
Scanner sc = new Scanner(System.in);
double balance;
double Amount;
do {
System.out.print("Enter your balance(must be greater than or
equal to 0) : ");
balance = sc.nextDouble();
} while (balance < 0);
boolean escape = true;
while (escape) {
menu();
int selector = sc.nextInt();
switch (selector) {
case 1:
System.out.println("How much amount you want to withdraw from
your account?");
System.out.print("Amount : ");
Amount = sc.nextDouble();
if (Amount < balance && Amount > 0) {
balance -= Amount;
System.out.println();
System.out.println("your new balance is: " + balance);
break;
} else if (Amount <= 0) {
check.check(Amount);
break;
}
else {
```

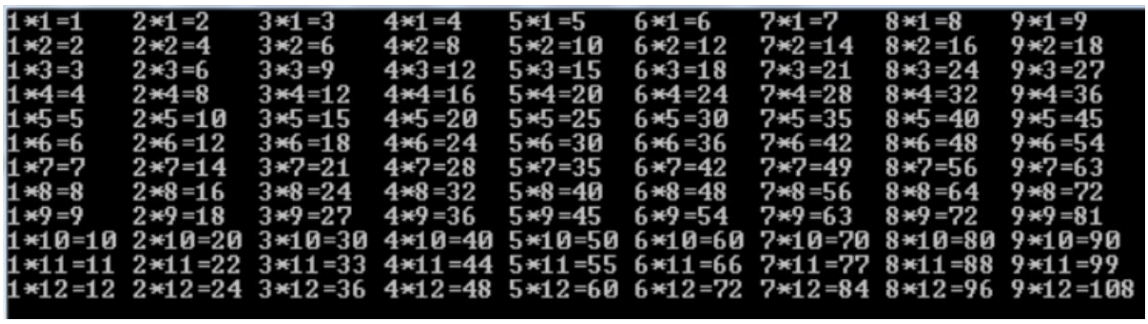
```
System.out.println("your balance is not sufficit to withdram "
+ Amount
+ " now you can withdraw lessthan or equal to yur balance,
deposit or exit!");
break;
}
case 2:
System.out.println("How much amount you want to deposit to
your account?");
Amount = sc.nextDouble();
if (Amount < 0) {
check.check(Amount);
break;
}
else {
balance += Amount;
System.out.println();
System.out.println("your new balance is: " + balance);
break;
}
case 3: {
System.out.println("your balance is : " + balance);
break;
}
case 4:
escape = false;
break;
}
}
}
}
```

OUTPUT

```
your new balance is: 100.0
--------------------------------------------------------------------------------
Select an option:
1. Withdraw Amount
2. Deposit Amount
3. Balance
4. Exit
--------------------------------------------------------------------------------
1
How much amount you want to withdraw from your account?
Amount : -12
invalid Amount!
please enter amount that is grtaer than 0
--------------------------------------------------------------------------------
Select an option:
1. Withdraw Amount
2. Deposit Amount
3. Balance
4. Exit
--------------------------------------------------------------------------------
2
How much amount you want to deposit to your account?
120

your new balance is: 220.0
--------------------------------------------------------------------------------
Select an option:
1. Withdraw Amount
2. Deposit Amount
3. Balance
4. Exit
```

2. Write a code that creates a multiplication table that looks like the one below. Using a 2–d array.



```
1*1=1   2*1=2   3*1=3   4*1=4   5*1=5   6*1=6   7*1=7   8*1=8   9*1=9
1*2=2   2*2=4   3*2=6   4*2=8   5*2=10  6*2=12  7*2=14  8*2=16  9*2=18
1*3=3   2*3=6   3*3=9   4*3=12  5*3=15  6*3=18  7*3=21  8*3=24  9*3=27
1*4=4   2*4=8   3*4=12  4*4=16  5*4=20  6*4=24  7*4=28  8*4=32  9*4=36
1*5=5   2*5=10  3*5=15  4*5=20  5*5=25  6*5=30  7*5=35  8*5=40  9*5=45
1*6=6   2*6=12  3*6=18  4*6=24  5*6=30  6*6=36  7*6=42  8*6=48  9*6=54
1*7=7   2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49  8*7=56  9*7=63
1*8=8   2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64  9*8=72
1*9=9   2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
1*10=10 2*10=20 3*10=30 4*10=40 5*10=50 6*10=60 7*10=70 8*10=80 9*10=90
1*11=11 2*11=22 3*11=33 4*11=44 5*11=55 6*11=66 7*11=77 8*11=88 9*11=99
1*12=12 2*12=24 3*12=36 4*12=48 5*12=60 6*12=72 7*12=84 8*12=96 9*12=108
```

Solution

```
import javax.swing.plaf.synth.SynthTreeUI;

class MultTable {
public static void main(String[] args) {
// create an empty 2-d array
int[][] table = new int[12][9];
// loop from 1 to 10
for (int i = 1; i <= 12; i++) {
// loop from 1 to 10
for (int j = 1; j <= 9; j++) {
// assign the product of i and j to the array element
table[i - 1][j - 1] = i * j;
}
}
// using nested for loop to display the result
// iterates down the table ie top to bottom(from 1 to 12)
System.out.println();
for (int i = 0; i < 12; i++) {
// iterates from right to left(from 1 to 9)
for (int j = 0; j < 9; j++) {
// this prints the value in the 2D array or table
System.out.print((j + 1) + "*" + (i + 1) + "=" + table[i][j] +
" ");
```

```
}
// this prints to new line after each iteration of i (the
outer for loop)
System.out.println();
}
}
}
```

OUTPUT

```
1*1=1     2*1=2     3*1=3     4*1=4     5*1=5     6*1=6     7*1=7     8*1=8     9*1=9
1*2=2     2*2=4     3*2=6     4*2=8     5*2=10    6*2=12    7*2=14    8*2=16    9*2=18
1*3=3     2*3=6     3*3=9     4*3=12    5*3=15    6*3=18    7*3=21    8*3=24    9*3=27
1*4=4     2*4=8     3*4=12    4*4=16    5*4=20    6*4=24    7*4=28    8*4=32    9*4=36
1*5=5     2*5=10    3*5=15    4*5=20    5*5=25    6*5=30    7*5=35    8*5=40    9*5=45
1*6=6     2*6=12    3*6=18    4*6=24    5*6=30    6*6=36    7*6=42    8*6=48    9*6=54
1*7=7     2*7=14    3*7=21    4*7=28    5*7=35    6*7=42    7*7=49    8*7=56    9*7=63
1*8=8     2*8=16    3*8=24    4*8=32    5*8=40    6*8=48    7*8=56    8*8=64    9*8=72
1*9=9     2*9=18    3*9=27    4*9=36    5*9=45    6*9=54    7*9=63    8*9=72    9*9=81
1*10=10   2*10=20   3*10=30   4*10=40   5*10=50   6*10=60   7*10=70   8*10=80   9*10=90
1*11=11   2*11=22   3*11=33   4*11=44   5*11=55   6*11=66   7*11=77   8*11=88   9*11=99
1*12=12   2*12=24   3*12=36   4*12=48   5*12=60   6*12=72   7*12=84   8*12=96   9*12=108
```

3. Write the program that display the following pattern

A.

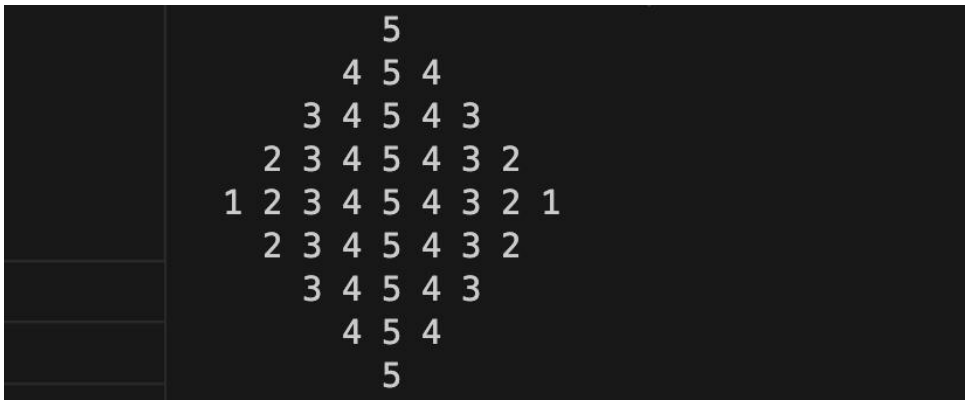
```
        5
      4 5 4
    3 4 5 4 3
  2 3 4 5 4 3 2
1 2 3 4 5 4 3 2 1
  2 3 4 5 4 3 2
    3 4 5 4 3
      4 5 4
        5
```

Solution

```
public class Pattern {
public static void main(String[] args) {
// The number of rows in the pattern
int n = 5;
// Loop for the upper half of the pattern
for (int i = n; i >= 1; i--) {
// Print spaces before the numbers
for (int j = 1; j < i; j++) {
System.out.print(" ");
}
```

```
// Print the numbers in descending order
for (int k = i; k <= n; k++) {
System.out.print(k + " ");
}
// Print the numbers in ascending order
for (int l = n - 1; l >= i; l--) {
System.out.print(l + " ");
}
// Move to the next line
System.out.println();
}
// Loop for the lower half of the pattern
for (int i = 2; i <= n; i++) {
// Print spaces before the numbers
for (int j = 1; j < i; j++) {
System.out.print(" ");
}
// Print the numbers in descending order
for (int k = i; k <= n; k++) {
System.out.print(k + " ");
}
// Print the numbers in ascending order
for (int l = n - 1; l >= i; l--) {
System.out.print(l + " ");
}
// Move to the next line
System.out.println();
}
}
}
```

OUTPUT



4. What is the difference and similarity between Java Applets Vs Java Application?

- ✓ Both of these are Java programs, but there is a significant difference between a Java application and a Java applet. The execution of the Java application begins with the main() method. On the other hand, the applet initializes through the init(). It does not use the main() method.
- ✓ The applications of Java are types of stand–alone programs that directly perform various general operations for their users. These do not require any APIs or browsers enabled by Java. Conversely, the applets serve as small programs that one can easily transmit across the internet. A web browser that is Java–compatible can execute these applets.
- ✓ The hardware or the operating system (OS) of the users' devices do not affect the Java applets. If the browser of the concerned system has a properly installed JVM, then it can run easily with the help of these JVMs. Also, the overall feel and look of the Java applications remain the same on various OS.

    - Java Applications are stand–alone programs that run directly on the underlying operating system with the support of a virtual machine. They do not require any web browser or HTML document to execute. They can access all kinds of resources available on the system, such as local files and networks. They start their execution with the main() method.
    - Java Applets are small programs that can be embedded into a web page and run inside a web browser or a special window called Appletviewer. They work on the client–side and can only access browser–specific services. They do not have access to the local system, such as files and networks. They start their execution with the init() method.
    - Both Java Applications and Java Applets are written in Java language and compiled with the javac command. They both use the same syntax and

libraries of Java. They both run on a virtual machine that provides platform independence and security.

Parameters	Java Application	Java Applet
Meaning and Basics	A Java Application is a type of program that can get independently executed on a computer.	A Java Applet is a small program that makes use of another application program so that we can execute it.
Main() Method	The execution of the Java application begins with the main() method. The usage of the main() is a prerequisite here.	The Java applet initializes through the init(). It does not require the usage of any main() method.
Execution	It cannot run alone, but it requires JRE for its execution.	It cannot run independently but requires APIs for its execution (Ex. APIs like Web API).
Installation	One needs to install a Java application priorly and explicitly on a local computer.	A Java applet does not require any prior installation.
Communication among other Servers	It is possible to establish communication with the other servers.	It cannot really establish communication with the other servers.
Read and Write Operations	The Java applications are capable of performing the read and write operations on various files present in a local computer.	A Java applet cannot perform these applications on any local computer.
Restrictions	These can easily access the file or data present in a computer system or device.	These cannot access the file or data available on any system or local computers.
Security	Java applications are pretty trusted, and thus, come with no security concerns.	Java applets are not very trusted. Thus, they require security.