**Advanced Programming – Midterm Exam 4 points each**

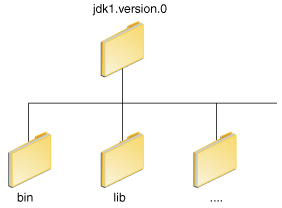
Name: **Madhava Bhima**

1. How does Java find the class files it needs for compilation and execution? (hint: which system variables would you set?)

**Answer:**

We have to set the PATH environment variable to run the executables (javac.exe, java.exe, javadoc.exe, and so on)

After installing the software, the JDK directory will have the structure shown below.



The bin directory contains both the compiler and the launcher.

We can run Java applications just fine without setting the PATH environment variable. Or, we can optionally set it as a convenience.

Set the PATH environment variable if you want to be able to conveniently run the executables (javac.exe, java.exe, javadoc.exe, and so on) from any directory without having to type the full path of the command. If you do not set the PATH variable, you need to specify the full path to the executable every time you run it, such as:

C:\Java\jdk1.7.0\bin\javac MyClass.java

The PATH environment variable is a series of directories separated by semicolons (;). Microsoft Windows looks for programs in the PATH directories in order, from left to right. You should have only one bin directory for the JDK in the path at a time (those following the first are ignored), so if one is already present, you can update that particular entry.

The following is an example of a PATH environment variable:

C:\Java\jdk1.7.0\bin;C:\Windows\System32\;C:\Windows\;C:\Windows\System32\Wbem

It is useful to set the PATH environment variable permanently so it will persist after rebooting. To make a permanent change to the PATH variable, use the **System** icon in the Control Panel. The precise procedure varies depending on the version of Windows and different platforms like Linux and Solaris.

The CLASSPATH variable is one way to tell applications, including the JDK tools, where to look for user classes. (Classes that are part of the JRE, JDK platform, and extensions should be defined through other means, such as the bootstrap class path or the extensions directory.)

The preferred way to specify the class path is by using the -cp command line switch. This allows the CLASSPATH to be set individually for each application without affecting other applications. Setting the CLASSPATH can be tricky and should be performed with care.

The default value of the class path is ".", meaning that only the current directory is searched. Specifying either the CLASSPATH variable or the -cp command line switch overrides this value.

To check whether CLASSPATH is set on Microsoft Windows NT/2000/XP, execute the following:

C:> echo %CLASSPATH%

On Solaris or Linux, execute the following:

% echo $CLASSPATH

If CLASSPATH is not set you will get a CLASSPATH: Undefined variable error (Solaris or Linux) or simply %CLASSPATH% (Microsoft Windows NT/2000/XP).

To modify the CLASSPATH, use the same procedure we used for the PATH variable shown above.

1. What is the output of the following ?

double x = 18.456

System.out.printf(“The number is : %5.3f”, x);

**Answer:**

Output:

The Number is : 18.456

Code:

**public** **class** sample {

**public** **static** **void** main(String[] args) {

**double** x = 18.456;

System.***out***.printf("The Number is : %5.3f",x);

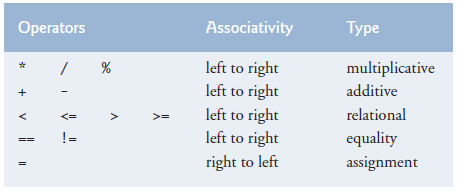
}

}

1. Give the precedence chart on page 60, Fig. 2.16 in the textbook, please find the value of variable “Z” below in each case:
2. float Z = 3 \* 14 % 7 + 8 / 2 – 1;
3. float Z = 2 \* ((8+1)\*5) / (7+2) – 4;

**Answer:**

The below figure shows the Operators Precedence and associativity



1. float Z = 3 \* 14 % 7 + 8 / 2 – 1;

Step1: Z = 3 \* 14 % 7 + 8 / 2 – 1; (multiplication)

3 \* 14 is **42**

Step2: Z = 42 % 7 + 8 / 2 – 1; (modulus)

42 % 7 is **0**

Step3: Z = 0 + 8 / 2 – 1; (division)

8 / 2 is **4**

Step4: Z = 0 + 4 – 1; (addition)

0 + 4 is **4**

Step4: Z = 4 – 1; (subtraction)

Step5: Z = 3;

1. float Z = 2 \* ((8+1)\*5) / (7+2) – 4;

Parentheses are used to group terms in Java expressions in the same manner as in algebraic expressions and it has higher precedence.

If an expression contains nested parentheses, the expression in the innermost set of parentheses is evaluated first.

Step1: Z = 2 \* ((8+1)\*5) / (7+2) – 4;(inner most parentheses expression)

(8+1) is **9**

Step2: Z = 2 \* (9 \* 5) / (7+2) – 4; (left most parentheses expression)

(9 \* 5) is **45**

Step3: Z = 2 \* 45 / (7+2) – 4; (parentheses expression)

(7+2) is **9**

Step4: Z = 2 \* 45 / 9 – 4; (multiplication)

2 \* 45 is **90**

Step5: Z = 90 / 9 – 4; (division)

90 / 9 is **10**

Step6: Z = 10 – 4; (subtraction)

Step7: z = 6;

Answer questions 4 & 5 given code below:

public class myTest {

private float balance;

public myTest( )

{

balance = 0;

}

public myTest( float x)

{

Balance = x;

}

}

1. Write a setter and a getter method for instance variable “balance”.

**Answer:**

Setter and getter methods for instance variable “balance” as shown below.

**public** **float** getBalance() {

**return** balance;

}

**public** **void** setBalance(**float** balance) {

**this**.balance = balance;

}

**myTest Class with getter and setter methods :**

**public** **class** myTest {

**private** **float** balance;

**public** myTest() {

balance = 0;

}

**public** myTest(**float** x) {

balance = x;

}

**public** **float** getBalance() {

**return** balance;

}

**public** **void** setBalance(**float** balance) {

**this**.balance = balance;

}

}

1. Class “myTest” has two constructors.

a) Write code to show how to instantiate “myTest” using each of these constructors.

b) For each use of constructor, what will the value of class variable “balance” be?

**Answer:**

1. Write code to show how to instantiate “myTest” using each of these constructors.

myTest myTest1 = **new** myTest(); // default constructor

myTest myTest2 = **new** myTest(3.456f); // constructor with float number

**Code:**

**public** **class** myTestDriver {

**public** **static** **void** main(String[] args) {

myTest myTest1 = **new** myTest(); // default constructor

myTest myTest2 = **new** myTest(3.456f); // constructor with float number

}

}

b) For each use of constructor, what will the value of class variable “balance” be?

**Output:**

The value of balance is :0.0

The value of balance is :3.456

**Code:**

**myTestDriver.java**

**public** **class** myTestDriver {

**public** **static** **void** main(String[] args) {

myTest myTest1 = **new** myTest(); // default constructor

myTest myTest2 = **new** myTest(3.456f); // constructor with float number

myTest1.displayBalance();

myTest2.displayBalance();

}

}

**myTest.java**

**public** **class** myTest {

**private** **float** balance;

**public** myTest() {

balance = 0;

}

**public** myTest(**float** x) {

balance = x;

}

**public** **float** getBalance() {

**return** balance;

}

**public** **void** setBalance(**float** balance) {

**this**.balance = balance;

}

**public** **void** displayBalance(){

System.***out***.println("The value of balance is :" + balance );

}

}

1. What is the main difference between while( ) and do…while( ) loops? Show with a brief example.

**Answer:**

In the while, the program tests the loop-continuation condition at the beginning of the loop, before executing the loop’s body; if the condition is false, the body never executes.

The do…while statement tests the loop-continuation condition after executing the loop’s body; therefore, the body always executes at least once. When a do…while statement terminates, execution continues with the next statement in sequence.

**Syntax:**

while (condition ){

Statement 1;

Statement 2;

Statement 3;

……

……

Statement n;

}

do

{

Statement 1;

Statement 2;

Statement 3;

……

……

Statement n;

} while (condition);

**Code Examples:**

*Calculate.java*

//Calculate the sum of the integers from 1 to 10

**public** **class** Calculate {

**public** **static** **void** main(String[] args) {

**int** sum = 0;// declare and initialize sum to 0 for totaling

**int** x = 1; // declare and initialize x to 1 for counting

**while** (x <= 10) // while x is less than or equal to 10

{

sum += x; // add x to sum

++x; // increment x

} // end while

System.***out***.printf("The sum is: %d\n", sum);

} // end main

} // end class Calculate

*DisplayEvenNumbers.java*

// To display all Even numbers for the given input number from user

**import** java.util.Scanner;

**public** **class** DisplayEvenNumbers {

**public** **static** **void** main(String[] args) {

// create Scanner to obtain input from command window to display Even numbers

Scanner input = **new** Scanner(System.***in***);

System.***out***.println("Enter the Number : "); // prompt the message

**int** maxNumber = input.nextInt();

**int** counter = 2; // initial even number

**do** {

System.***out***.println(counter);

counter += 2;

} **while** (counter <= maxNumber);

}

}

1. What is the output of the following code?

int x = 4, y = 100;

System.out.println(“Hello!”);

if( x > 5 )

if( y > 5 )

System.out.println(“x and y are > 5”);

else

System.out.println(“x is <= 5”);

System.out.println(“Betty!”);

**Answer:**

*Output:*

Hello!

Betty!

**Code:**

**public** **class** DanglingElseProblem {

**public** **static** **void** main(String[] args) {

**int** x = 4, y = 100;

System.***out***.println("Hello!");

**if**( x > 5 )

**if**( y > 5 )

System.***out***.println("x and y are > 5");

**else**

System.***out***.println("x is <= 5");

System.***out***.println("Betty!");

}

}

1. What is the output of the following code?

int x = 3, y = 100;

while( x > 0)

{

if( y < 99 )

break;

System.out.println(“I got it!”);

x--;

y--;

}

System.out.println(“And it is all good!”);

**Answer:**

I got it!

I got it!

And it is all good!

**Code:**

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

**int** x = 3, y = 100;

**while**( x > 0)

{

**if**( y < 99 )

**break**;

System.***out***.println("I got it!");

x--;

y--;

}

System.***out***.println("And it is all good!");

}

}

1. What is the output of the following code? Is this an infinite loop? Why?

int m = 5, n = 80;

while( m < 10)

{

System.out.println(“ n = “ + n);

n = n + 10;

if( n < 100 )

continue;

else

break;

}

**Answer:**

n = 80

n = 90

It is not infinite loop, while condition always true but we break the while loop repetition by checking if (n < 100) condition is false in second iteration and hence it break the while repetition loop in else block using the keyword “break”.

1. What is the output of the following code?

int x = 5;

System.out.println( --x );

System.out.println( x++ );

System.out.println( x );

**Answer:**

4

4

5

1. What is the output of the following code?

int x = 10;

M1(x); //call the method M1

System.out.println(“x4 = “ + x);

public void M1(int y) // a method inside the same class

{

y = 200;

System.out.println(“y5 = “ + y);

}

**Answer:**

y5 = 200

x4 = 10

1. What is the output of the code below?

int K = 4;

for( int i=0; i<k; i++)

{

System.out.println(“Blue”);

}

**Answer:**

Blue

Blue

Blue

Blue

1. What is the output of the code below?

int K = 2;

for( int i=5; i>k; i--)

{

System.out.println(“Red”);

}

**Answer:**

Red

Red

Red

What is the problem with the code snippets 14 to 19? (hint: be sure to check if it is infinite loop)

1. int counter = 1;

do {

System.out.println(“Yellow”);

counter++;

} while(counter < 5)

**Answer:**

Syntax error on token ")", ; expected after this token

If we put “ ; ” at the end of while condition in the above code snippet then we will get the below output:

Yellow

Yellow

Yellow

Yellow

1. Given: int k = 17; int p = 0;

switch(k)

{

case 9:

case 10:

p = 5;

default:

p = 11;

}

**Answer:**

The default case will execute and value “11” will assign into p variable. If we print p value then it will print 11.

int z = 5, sum = 0;

while( z >= 0 )

sum += z;

**Answer:**

It is an infinite loop because while condition always true.

To avoid this infinite loop error in the above code snippet, we can decrement “z” value inside while loop so that while condition will return false after some iterations.



int x = 60, total = 0;

while ( x <= 100 )

total += x;

x++;

**Answer:**

It is an infinite loop because while condition always true. Here we increment x value by 1 using postfix operator, but that statement never execute because it is not part of while loop. Since there is no { } curly braces after while condition it will execute the first statement after while condition and it never reach to x++ statement.



for(int i = 0; i <= 10; )

System.out.println(i);

**Answer:**

It is an infinite loop and it prints value “0” infinitely, to avoid this infinite loop either we can increment “i” value in for header or inside for loop block.



int Age = 29;

char gender = ‘F’;

int count = 0;

if( Age < 35 & gender == ‘F’ )

count++;

**Answer:**

It will execute without any error. And if condition will return “true” and count variable increment by 1.

The **boolean logical AND (&) and boolean logical inclusive OR (|)** operators are identical to the && and || operators, except that the & and | operators always evaluate both of their operands (i.e., they do not perform short-circuit evaluation). So, the expression

( gender == 1 ) & ( age >= 65 )

evaluates age >= 65 regardless of whether gender is equal to 1. This is useful if the right operand of the boolean logical AND or boolean logical inclusive OR operator has a required **side effect**—a modification of a variable’s value. For example, the expression

( birthday == true ) | ( ++age >= 65 )

guarantees that the condition ++age >= 65 will be evaluated. Thus, the variable age is incremented,

regardless of whether the overall expression is *true* or *false*.

1. Explain with examples what is “short-circuit evaluation” for logical operators “&&” and “||”.

**Answer:**

The parts of an expression containing && or || operators are evaluated only until it’s known whether the condition is true or false. Thus, evaluation of the expression

( gender == FEMALE ) && ( age >= 65 )

stops immediately if gender is not equal to FEMALE (i.e., the entire expression is false) and continues if gender is equal to FEMALE (i.e., the entire expression could still be true if the condition age >= 65 is true). This feature of conditional AND and conditional OR expressions is called short-circuit evaluation.