|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| USN |  |  |  |  |  |  |  |  |  |  |

 |  |
| Internal Assessment Test 1 – Sept. 2018 |
| Sub: | Programming in JAVA | Sub Code: | 15CS561 | Branch: | ECE/EEE |
| Date: | 10-09-2018 | Duration: | 90 min’s | Max Marks: | 50 | Sem / Sec: | 5 (ALL SEC) | OBE |
| Answer any FIVE FULL Questions | MARKS | CO | RBT |
| 1 (a) | Discuss three OOP principles. | [2+2+2] | CO1 | L1 |
|  (b) |  Explain steps to execute simple java program and role of JVM in execution.  | [2+2] | CO1 | L2 |
| 2 (a) | Explain scope and lifetime of variables with an example. | [5] | CO2 | L1 |
|  (b) |  What is narrowing and widening. Explain with an example. | [5] | CO2 | L2 |
| 3 | Explain how array in java works differently than C(syntax and examples are compulsory). Write java program to display0 1 2 3 45 6 7 8 9 | [5+5] | CO2 | L3 |
| 4 (a) | Explain Ternary operator. Write a java program to display largest of three numbers using ternary operator. | [02+04] | CO2 | L3 |
|  (b) | Evaluate i)3+2%10&&2 ii) 2|1-!10\*++5 | [02+02] | CO2 | L2 |
| 5 (a) | Explain if, if..else, nested if statements with syntax and suitable example. | [2+2+2] | CO2 | L2 |
|  (b) | Explain use of for..each loop with suitable example. Write same example using simple for loop. | [2+2] | CO2 | L2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 (a) | Demonstrate use of i) break statement in do..while loop ii) continue statement in while loop with examples.  | [3+3] | CO2 | L3 |
|  (b) | Explain use of short circuit property with example. | [4] | CO2 | L2 |
| 7 (a) | Write program to display Prime numbers between 1 to 100.  | [5] | CO2 | L3 |
|  (b) | Write a program to check given year is leap year or not.[take year=2020] | [5] | CO2 | L3 |

SCHEME

|  |  |  |  |
| --- | --- | --- | --- |
| **Question #** | **Description** | **Marks Distribution** | **Max Marks** |
| 1 a |  Each OOP principal  |  2 M2 M2M |  6M |  6M |
| 1 b |  steps to execute simple java program  role of JVM |  2 M2 M |  4M |  4M |
| 2a |  scope and lifetime of variables with an example |  4+1 M |  5M |  5M |
| 2b |  narrowing and widening of expression . Explain with an example. |  4+1 M |  5M |  5M |
| 3  |  array in java works differently than C(syntax and examples are compulsory).Program |  4+1 M5M |  10M |  10M |
| 4 a |  Ternary operator. Write a java program to display largest of three numbers using ternary operator. |  2M4M |  6M |  6M |
| 4 b |  i)3+2%10&&2 ii) 2|1-!10\*++5 |  2 M2 M |  4M |  4M |
| 5 a | if,if..else, nested if statementswith syntax and suitable example. | 2M2M2M  | 6M | 6M |
| 5 b | For each loopExample | 2M2M |  4M |  4M |
| 6a | i) break statement in do..while loopii) continue statement in while loop with examples. | 3M3M | 6M | 6M |
| 6 b | short circuit property with example. | 4M |  4M |  4M |
| 7a | program to display Prime numbers between 1 to 100.  | 5M  | 5M | 5M |
| 7 b | program to check given year is leap year or not. | 5M  | 5M | 5M |
|  |  |  |  |  |  |

|  |  |
| --- | --- |
| 1) A)Discuss three OOP principles. | [2+2+2] |

**Encapsulation:** Encapsulation can be defined as the procedure of casing up of codes and their associated data jointly into one single component.

In simple terms, encapsulation is a way of packaging data and methods together into one unit. Encapsulation gives us the ability to make variables of a class keep hidden from all other classes of that program or namespace.

 Hence, this concept provides programmers to achieve data hiding. Programmers can have full control over what data storage and manipulation within the class

**Inheritance:** Inheritance can be defined as the procedure or mechanism of acquiring all the properties and behavior of one class to another, i.e., acquiring the properties and behavior of child class from the parent class.

Java supports three types of inheritance. These are:



**Polymorphism:** The word polymorphism means having multiple forms. The term Polymorphism gets derived from the Greek word where poly + morphos where poly means many and morphos means forms.

· Static Polymorphism

· Dynamic Polymorphism.

b) Explain steps to execute simple java program and role of JVM in execution [2+2]

Steps:

a) After typing the program in the terminal we have to type javac progranmane.java and hit enter

b) Then again in the terminal (if no error is there) the we have to type java program name and hit enter key

Role of JVM in execution can be described as follows:



· Reading Bytecode.

· Verifying bytecode.

Linking the code with the library

2)a) Explain scope and lifetime of variables with an example [5]

· Java allows variables to be declared within any block. A block defines a scope. Thus, each time you start a new block, you are creating a new scope.

· A scope determines what objects are visible to other parts of your program. It also determines the lifetime of those objects

· In Java, the two major scopes are:

i) defined by a class and

ii) those defined by a method.

· As a general rule, variables declared inside a scope are not visible (that is, accessible) to code that is defined outside that scope. Thus, when we declare a variable within a scope, we are localizing that variable and protecting it from unauthorized access and/or modification.

· Each time you create a block of code, you are creating a new, nested scope. When this occurs, the outer scope encloses the inner scope. This means that objects declared in the outer scope will be visible to code within the inner scope. However, the reverse is not true. Objects declared within the inner scope will not be visible outside it.

// Demonstrate block scope.

class Scope {

public static void main(String args[]) {

int x; // known to all code within main

x = 10;

if(x == 10) { // start new scope

int y = 20; // known only to this block

// x and y both known here.

System.out.println("x and y: " + x + " " + y);

x = y \* 2;

}

// y = 100; // Error! y not known here

// x is still known here.

System.out.println("x is " + x);

}

}

· Here is another important point to remember: variables are created when their scope is entered, and destroyed when their scope is left.

· If a variable declaration includes an initializer, then that variable will be reinitialized each time the block in which it is declared is entered.

b) What is narrowing and widening. Explain with an example. [5]

**Widening :**When one type of data is assigned to another type of variable, an automatic type conversion will take place if the following two conditions are met:

• The two types are compatible.

• The destination type is larger than the source type.

When these two conditions are met, a widening conversion takes place. For example, the int type is always large enough to hold all valid byte values, so no explicit cast statement is required. For widening conversions, the numeric types, including integer and floating-point types, are compatible with each other.

 **Narrowing:** In case of stornt int value to a byte variable, conversion will not be performed automatically, because a byte is smaller than an int. This kind of conversion is sometimes called a narrowing conversion, To create a conversion between two incompatible types, we must use a cast. A cast is simply an explicit type conversion. It has this general form:

(target-type) value

For example, the following fragment casts an int to a byte. If the integer’s value is larger than the range of a byte, it will be reduced modulo (the remainder of an integer division by the) byte’s range.

int a;

byte b;

// ...

b = (byte) a;

3 ) Explain how array in java works differently than C(syntax and examples are compulsory). Write java program to display

 0 1 2 3 4

 5 6 7 8 9

 [5+5]

In C programming declaration of array is as follows:

 Datatype arr-name[size of the array];

 int arr[20];

But in array is declared as follows:

 Datatype arr-name[ ]= new Datatype [size of the array];

 Int arr[ ]= new arr[20];

 In java array will be declared with the keyword “new” and for that all the elements in the array will be initialized to 0. But in C as no new keyword is used at the time of array declaration so elements of array in C at the time of declaration will be initialized to null.

java program to display

 0 1 2 3 4

 5 6 7 8 9

**public** **class** display {

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

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

 {

 System.***out***.print(i+" ");

 **if**(i==4)

 {

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

 }

 }

 }

}

4) a) Explain Ternary operator. Write a java program to display largest of three numbers using ternary operator [5]

· ternary (three-way) operator that can replace certain types of if-then-else statements.

· This operator is the ?.

· The ? has this general form: expression1 ? expression2 : expression3

· Here, expression1 can be any expression that evaluates to a boolean value. If expression1 is true, then expression2 is evaluated; otherwise, expression3 is evaluated.

· The result of the ? operation is that of the expression evaluated.

· Both expression2 and expression3 are required to return the same type, which can’t be void.

Largest among three numbers:

**public** **class** Ternary {

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

 **int** a=40,b=39,c=99,res;

 res=(a>b)?((a>c)?a:c):((b>c)?b:c);

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

 }

}

4b>

i)3+2%10&&2

 3+2&&2

 5&&2

1

ii) 2|1-!10\*++5

3-0\*6

3

5a> Explain if, if..else, nested if statements with syntax and suitable example.

An if statement consists of a Boolean expression followed by one or more statements.

Syntax

Following is the syntax of an if statement −

if(Boolean\_expression) {
 // Statements will execute if the Boolean expression is true
}

If the Boolean expression evaluates to true then the block of code inside the if statement will be executed. If not, the first set of code after the end of the if statement (after the closing curly brace) will be executed.

Flow Diagram



Example

public class Test {

 public static void main(String args[]) {
 int x = 10;

 if( x < 20 ) {
 System.out.print("This is if statement");
 }
 }
}

If..else

An if statement can be followed by an optional else statement, which executes when the Boolean expression is false.

Syntax

Following is the syntax of an if...else statement −

if(Boolean\_expression) {
 // Executes when the Boolean expression is true
}else {
 // Executes when the Boolean expression is false
}

If the boolean expression evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

Flow Diagram



Example

public class Test {

 public static void main(String args[]) {
 int x = 30;

 if( x < 20 ) {
 System.out.print("This is if statement");
 }else {
 System.out.print("This is else statement");
 }
 }
}

Nested-IF

It is always legal to nest if-else statements which means you can use one if or else if statement inside another if or else if statement.

## Syntax

The syntax for a nested if...else is as follows −

if(Boolean\_expression 1) {
 // Executes when the Boolean expression 1 is true
 if(Boolean\_expression 2) {
 // Executes when the Boolean expression 2 is true
 }
}

You can nest else if...else in the similar way as we have nested *if* statement.

## Example

public class Test {

 public static void main(String args[]) {
 int x = 30;
 int y = 10;

 if( x == 30 ) {
 if( y == 10 ) {
 System.out.print("X = 30 and Y = 10");
 }
 }
 }
}

5b>Explain use of for..each loop with suitable example. Write same example using simple for loop.

### Syntax

Following is the syntax of enhanced for loop −

for(declaration : expression) {
 // Statements
}

* Declaration − The newly declared block variable, is of a type compatible with the elements of the array you are accessing. The variable will be available within the for block and its value would be the same as the current array element.
* Expression − This evaluates to the array you need to loop through. The expression can be an array variable or method call that returns an array.

### Example

public class Test {

 public static void main(String args[]) {
 int [] numbers = {10, 20, 30, 40, 50};

 for(int x : numbers ) {
 System.out.print( x );
 System.out.print(",");
 }
 System.out.print("\n");
 String [] names = {"James", "Larry", "Tom", "Lacy"};

 for( String name : names ) {
 System.out.print( name );
 System.out.print(",");
 }
 }
}

Using simple for loop

public class Test {

 public static void main(String args[]) {
 int [] numbers = {10, 20, 30, 40, 50};

 for(int x : numbers ) {
 System.out.print( x );
 System.out.print(",");
 }
 System.out.print("\n");
 String [] names = {"James", "Larry", "Tom", "Lacy"};

 for( int i=0;i<4;i++ ) {
 System.out.print( name[i] );
 System.out.print(",");
 }
 }
}

6a>

Demonstrate use of i) break statement in do..while loop ii) continue statement in while loop with examples.

The break statement can be used with any of Java’s loops, including intentionally

infinite loops. For example, here is the preceding program coded by use of a while loop.

The output from this program is the same as just shown.

// Using break to exit a while loop.

class BreakLoop2 {

public static void main(String args[]) {

int i = 0;

do{

if(i == 10) break; // terminate loop if i is 10

System.out.println("i: " + i);

i++;

}while(i < 100);

System.out.println("Loop complete.");

}

}

In while and do-while loops, a continue statement

causes control to be transferred directly to the conditional expression that controls the loop

For all three loops, any intermediate code is bypassed.
Here is an example program that uses continue to cause two numbers to be printed on
each line:
// Demonstrate continue.
class Continue {
public static void main(String args[]) {

Int i=0;
while( i<10)

{
System.out.print(i + " ");
if (i%2 == 0) continue;
System.out.println("");

i++ ;
}
}
}
This code uses the % operator to check if i is even. If it is, the loop continues without printing
a newline. Here is the output from this program:
0 1
2 3
4 5
6 7
8 9

**6b>Explain use of short circuit property with example.**

These are secondary versions of the Boolean AND and OR operators, and are known as

short-circuit logical operators. As you can see from the preceding table, the OR operator

results in true when A is true, no matter what B is. Similarly, the AND operator results in

false when A is false, no matter what B is.

**a=10 b=0**

**if(b&&a) here since b is 0; result is going to be 0; no need to check a**

**if(a||b) here since a is non zero; result is going to be 1; no need to check b**

**7a>Write program to display Prime numbers between 1 to 100.**

**class PrimeNumbers
{
 public static void main (String[] args)
 {
 int i =0;
 int num =0;
 //Empty String
 String primeNumbers = "";

 for (i = 1; i <= 100; i++)
 {
 int counter=0;
 for(num =i; num>=1; num--)
 {
 if(i%num==0)
 {
 counter = counter + 1;
 }
 }
 if (counter ==2)
 {
 //Appended the Prime number to the String
 primeNumbers = primeNumbers + i + " ";
 }
 }
 System.out.println("Prime numbers from 1 to 100 are :");
 System.out.println(primeNumbers);
 }
}**

**7b>Write a program to check given year is leap year or not.[take year=2020]**

**import java.util.Scanner;**

**public class Check\_Leap\_Year**

**{**

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

 **{**

 **int year = 400;**

 **boolean flag = false;**

 **if(year % 400 == 0)**

 **{**

 **flag = true;**

 **}**

 **else if (year % 100 == 0)**

 **{**

 **flag = false;**

 **}**

 **else if(year % 4 == 0)**

 **{**

 **flag = true;**

 **}**

 **else**

 **{**

 **flag = false;**

 **}**

 **if(flag)**

 **{**

 **System.out.println("Year "+year+" is a Leap Year");**

 **}**

 **else**

 **{**

 **System.out.println("Year "+year+" is not a Leap Year");**

 **}**

 **}**

**}**