module 1 - 16

module 2 - 18

module 3 - 18

module 4 - 18

module 5 - 05

Module 2

* Code section - 6 to 8 question
* What are the features of java?
* How can we compile java code?
* Who is platform independent?

java language or jvm?java langauage or jdk?

* What is class loader,jit?
* What is java fundamental?

Int a=10;

Int $a=10;

Int \_a=10;

Int a10=10;

Int la=10;

Int #a=10;

* Int a = 1 & 5;

Int = 1/5;

* What is static variable?

Which have only one copy for entire object of the same class .We don’t need to create the object for static variable. We can use instead class name for calling the static object

For ex:

Public class employee{

Static int pf;

}

Public Class test{

Main(){

Employee.pf;

Println(employee.pf);

}

* Static method

Static method only uses the static variable.

We cannot call the non static method into a static method without making any object.

We can call the static method in non-static method as we don’t need to create the object.

Public class test{

Static int a;

Public static void main(String[] args){

Int temp = 10; //local static

Println(temp);

}

* Static Block:

It is executed before the main method.

* Local variable

Inside some method we write the local method.

For ex:

Public void getData(){

Int a = 10;

}

Instance vaiable has scope for entire class and it has some default value.

Local variable has the scope only for the particular method and it has no default values we have to initialize the value;

* Packages:

Default package is java.lang

We use packages for avoiding the naming ambiguity.

* Import static

import static java.lang.Math.\*;

* this reference:

used to refer current object from any method or constructor.

it can be written as **this()**

* Constructor:

If a parameterized constructor is written in class then we have to create the default constructor.

* Object class:

Equals,void finalize(),class getclasss()

* Casting Between Primitive Types:

Changing the data type based on their increasing order;

Byte-Short-long-Int-float-double

* String Handling:

String is immutable

String buffer is mutable and thread safe

String builder is mutable but not thread safe and faster then string buffer.

* Super:

In child class the super c

Key word will call the default constructor of the parent class.

* Overloading:

Same method name, different parameters and in the same class.

* Abstract Class:

Atleast One abstract method and any number of simple methods which does not have any body

Abstract class is created only for inheritance

Object of abstract class cannot be created

Public void abstract ….

Extends is used for abstract class

* Interface

It is 100% abstract

All the methods in the interface are abstract

Implements is used for the interface

* Exceptions

**Checked/Compile time exception**

This type of exception is checked while compiling.

These are inherited from the core Java class Exception.

Should be handled in the code.

IOException, FileNotFound

**Unchecked/Runtime Exception**

This is at the runtime exception.

The code handles this exception explicitly.

NullPointerException

* The try catch final method

In multiple catch the super class that is catch(Exception e){} is always written at the last.

* Throw and throws:

**Throws:**

This describes the exceptions which can be raised by a method.

**Throw:**

This raises an exception to the first available handler in the call stack

* Arrays:

Ex:

Public class Test{

Static int a;

Static Boolean b[0];

Main(){

Println(b);

}

}

Ans: Null pointer Exception

Ex 2:

Public class Test{

Static int a;

Static Boolean b[];

Main(){

Println(b);

}

}

Ans: Null

* Why to use generic?

**1) Type-safety :** We can hold only a single type of objects in generics. It doesn’t allow to store other objects.

**2) Type casting is not required:** There is no need to typecast the object.

**3) Compile-Time Checking:** It is checked at compile time so problem will not occur at runtime. The good programming strategy says it is far better to handle the problem at compile time than runtime.

* Different streams

Byte stream

Character stream

Buffered stream

Data stream

Object stream

Scanning and formatting

* Which annotation is used for test?

@test

What are the appenders in log4j

Priority levels in log4j

Thread Life cycle

Different ways to create thread ??

Lamba function annotation

@FunctionalInterface