1. Why do we need a programming language?

Computer eken wadak karaganna awashya instructions labadenna

1. Key features of java

Platform Independence

Object-Oriented

Multi-threading

Static type ()

1. Oop concept 4

1. Encapsulation - data single unit ekakata wrap kereema

2. Inheritance

3. Polymorphism

4. Abstraction

1. Real world examples

1. Encapsulation -

2. Inheritance -

3. Polymorphism - eka manussayekge eka eka forms

4. Abstraction -

1. Abstract class vs interface

Abstract class - concrete method thiyeno

1. JVM

Byte code eka run kereema.

1. JRE , JDK , JVM

1. Compiler , Java Interpreter, JIT(Just in time) Compiler

===

1. Access modifiers

protected

same class

same package

different package subclass access පුලුවන්. (Through the inheritance only)

different package non-subclass වල access නෑ.

public

ඕන තැනක access කරන්න පුලුවන්.

privet

same class විතරයි access කරන්න පුලුවන්.

default

same class, same package විතරයි access කරන්න පුලුවන්.

1. final, finally, finalize()

final - class ekkata dammoth extend karanna ba (Sting class)

- method ekakata dannot

Finally - try catch wala try , catch block deken passe aniwaryen run wena block eka, return tibbat finally eka run wela return wenne.

finalize()- object class eka atule thiyenne, hadana object ekak garbage collector ekata eligible unama eeka arn yano.

1. String

[String API](https://docs.google.com/document/d/1x_OILLbmuCumnU0wAd6rgEOxPcuvMSzXpJTLonqslAY/edit#heading=h.na2pijn6e05s)

1. Errors vs exception

Superclass throwable

Try catch, trow

Main method eken eliyata exception trow karaot - uncoach exception handler program eka terminate karano, print stack trace

Check - compile time error

uncheck - run time error

1. Collection framework

[Collection Framework](https://docs.google.com/document/d/1vcvI0DAtFxIvAvgrvcnfcOtB7z3ScufNYAUl6sm2ySQ/edit)

1. Lambda expression (eka method ekak thiyena interface ekak thiyenna oona) (functional interface)

(parameter\_list) -> { body } ,, listners

1. Stream API
2. Garbage collector - destroy unreferenced (garbage collector eligible) objects from memory
3. Threads

[MultiThread](https://docs.google.com/document/d/1UPXbpfI4k-zoOaCYrpIeBVPmqJQL9Q39VbI8IOmabNA/edit#heading=h.yyn90f8z7s9z) .

wait(), notify(), and notifyAll() <- object class ekae thiyenne

wait() <- Thread eka wait karano

notify() <- wait eke inna thred eka wake karano

notifyAll() <- okkoma wake karano

1. [Generics](https://docs.google.com/document/d/16mYwmcoNXeGdLGHC-IyMoabT_2m70n_7P79_YSFzm3w/edit)

1. Autoboxing and unboxing

int primitiveInt = 10;

Integer wrappedInt = primitiveInt; // Autoboxing - int to Integer

Integer wrappedInt = 20;

int primitiveInt = wrappedInt; // Unboxing - Integer to int

1. this vs super

this - current object eka call kari

super - super class eka access karano

1. Serialization , Deserialization

Serialization <- object ekak byte stream ekaka karano

Deserialization <- byte stream ekak object ekak karano

Serializable Interface eken wenne. Marker interface ekak. Mokut na atule.

1. Java Reflections
2. switch vs if

switch - condition eka true unama okkoma run weno brack eka natnm

if - condition eka true nun adala block eka run weno

1. Input Stream Classes (socket programming wala use weno)

ObjectInputStream, InputStream,

1. static
2. Instance of

Object obj = "Hello";

if (obj instanceof String) {

System.out.println("obj is an instance of String");

} else {

System.out.println("obj is not an instance of String");

}

Object ekak ee class ekakema instance ekakda kiyala check karano.

1. constructor vs method
2. shallow copy vs deep copy

shallow copy <- address eka copy karaganno (fast)

Deep copy <- object eke copyak ganno

1. Marker interface vs functional interface

Marker interface <-

functional interface <-

1. Variable vs Array

Variable - value ekak . . .

Array - same type value keepayak memory eke temporary store karaganna use karana mechanism ekak

1. Why use prepared statements in JDBC connections?

Query ekakata dynamically values assign karanna puluwn, run time inject values

1. JIT compiler
2. JDBC transaction

setAutocommit(false) <- database eke table eka permanently update wenne na

1. System class

System ekata adaala input output … handle karano

1. Generics
3. Daemon thread

Background eke run wena threads.

volatile keyword

1. break vs continue
2. toString()
3. Array vs arraylist
4. Stack vs Queue
5. Conversion and casting

========

Spring

Hackerrank

Observer observable design patterns