Agenda

- Revision
- Strings
    - String
    - StringsBuffer
    - StringBuilder
- Enum
- Arrays class
- JVM Architecture

String (Demo01)

- Character is a wrapper class in java for the primitive type char
- char takes 2 bytes in the memory
- the sequence of characters is called as String
- String is a class in java
- The variable that we create of the String class is called as reference
- String is immutable
- If you want to use mutable Strings then use StringBuffer and StringBuilder class

StringBufffer and StringBuilder (Demo02)

- Both these classes are mutable classes
- StringBuffer is TreadSafe
- StringBuilder is not Threadsafe
- The object for these classes can be creates using new operator only
- hascode and equals method is not overidden inside these classes

Enum (Demo03)

- It is used to store constant values.
- It is generally used for making the code readable
- Most widely used for switch cases.
- We cannot extend the enum from the classes however we can impelment an interface
- enum is internally converted into class
- all the fields inside enum are converted as public static final field of the class

Arrays class (Demo04 -> Program01)

- It is a class in java.util package
- It has some helper methods that can be used over arrays of java
- eg
    - sort(arr)
    - equals(arr1,arr2)
    - binearySearch(arr, key)
    - toString(arr)

String Tokenizer (Demo04 -> Program02)

- It is a class defined in java.util package
- It is used to split the strings.
- It have only 3 constructors which can be used as per the requirement
    - StringTokenizer(String str)
    - StringTokenizer(String str, String delim)
    - StringTokenizer(String str, String delim, boolean returnDelims)
- the default split is done using the delimiter as " " (empty space)
- hasMoreTokens() is used to check for tokens(words) if present
- nextToken() is used to get the token(word)

JVM Architecture

- It consists of
    - 1. ClassLoader Subsytem
    - 2. Memory Areas
    - 3. Exeution Engine
- when the code is compiled using the tool javac it genereates .class file
- .class is used for the execution.
- we use the tool `java` for executiong .class file.
- when we execute the .class file JVM is invoked and .class file is given as a input to the JVM

1. Class Loader Subsytem

- It consists of 3 stages
    - 1. Class Loading
    - 2. Linking
    - 3. Initialization

1. Class Loading

- It consists of 3 class loaders
    - 1. Bootstrap classloader
        - Used to load all the java builtin classes from jre/lib jars (rt.jar)
    - 2. Extension classloader
        - used to load the extended classes from jre/lib/ext dir
    - 3. Application classloader
        - used to load the classes from the specified CLASSPATH

2. Linking

- It consists of 3 parts
    - 1. Verification
        - the byte code verifiifier verifies if the .class file are compiled by the valid java compiler or not.
    - 2. Preparation
        - All the static variables will get the memory and will be initialized with the default values.

    - 3. Resolution
        - All the symbolic references from the constant pool will be replaced by thier actual references.

3. Initialization

- It assigns all the static varaibles with the assigned values using the initializers(field initilizers)
- All the static blocks if present gets executed.

2. Memory Areas

- It consists of 5 differnt memory
    - 1. Method Area
    - 2. Heap Area
    - 3. Stack Area
    - 4. PC Registers
    - 5. Native Method Stack

1. Method Area

- It is the area used to store all the class methods (metadata of the class).
- All static varaibles gets space on this area.

2. Heap Area

- All the objects that are created using new operator are placed in this area.
- String Pool is also part of this heap area.

3. Stack Area

- For every thread in java a stack is created in this stack area
- When a method is called , a stack frame is created for the FAR of that method.
- When method executes the FAR gets destroyed and also the stackframe is removed automatically

4. PC Registers

- PC is created for every new Thread created in java.
- It points to the address of next instruction to be executed.
- When methods fetches the address of next instruction, the address inside PC gets incremented automatically

5. Native Method Stack

- It is also creating stack for every new thread.
- It is used to store the stackframes for the native methods

3. Execution Engine

- It consists of
    - 1. Interepretur

    - 2. JIT Compiler
    - 3. Garbage Collector

1. Interpretur

- It intrepretes the byte code into native/machine understandable code
- Every method is interpreted atleast once by this interpreter
- If a method is getting called multiple no of time then interpreting make the execution slower.
- to overcome this JIT compiler is used.

2. JIT Compiler

- It is a compiler that is used to compile the methods that gets called multiple time and cache it.
- The next time the same method is called it is not interpreted , however the compiled native form of that metehod is given from the JIT cache.

3. garbage Collector

- It is used to clear the unreferenced objects from the heap section.
- It gets invioke automatically as per the scheduling time and clears the heap memory from unused objects.