Day 2

- src.zip : It contains source code of java API.
- rt.jar : It contains compiled code of java API
- java docs : It contains documentation of java API.

Structure of Jar File

- Jar File Contains:
    - Manifest File
        - It contains metadata of jar file.
    - Package(s)
        - Sub Package
        - Interface
        - Class
        - Enum (Final class)
        - Error (class)
        - Exception (class)
        - Annotation Types (Interface)

Interface Can contain

1. Nested Interface
2. Constant / Final field
3. Abstract Method
4. Static Method
5. Default Method

Class Can contain

1. Nested Type(Enum, class, Interface)
2. Field
3. Constructor
4. Method

C++ vs Java Terminology

C++ Terminology java Terminology

1. class : class
2. object : instance
3. Data Member : field
4. Member Function : method
5. Access Specifier : access modifier
6. Base class : super class
7. Derived class : sub class
8. Derived From : extends from

9. Pointer : reference

- Concrete class : We can instantiate concrete class.
- Abstract class : We can instantiate abstract class.
- Final class : It is a concrete class that we can not extend
- Final Method : We can not override final method
- Final field : We can not modify state of final variable.

Exploring rt.jar

- rt.jar file contains 4 main packages

1. com
2. java
3. javax
4. org

- java main package contains 14 sub packages

1. applet 2. awt 3. beans 4. io
2. lang 6. math 7. net 8. nio
3. rmi 10. security 11. sql 12. text
4. time 14. util

- In java, to access sub package, dot operator is used.

- java.lang package contains all the fundamental classes of core java.

    1. System
    2. Object, Class
    3. Wrapper Classes
    4. String Classes :
        - String
        - StringBuffer
        - StringBuilder
    5. Thread management Classes
    6. Math class

Execution Flow

- java copiler generates .class file. It contains compiled code i.e Bytecode.
- Command to see Bytecode: javap -c Program.class
- Bytecode is object oriented assembly language code designed for JVM.
- A code which is designed to run by virtual machine is called managed code
- In MS.NET, MSIL code and In java Bytecode is called managed code.

System.out.println

- System is final class declared in java.lang package.

```
package java.lang
public final class System extends Object
{
    //Fields of System class
    public static final InputStream in;public static final PrintStream
out;
    public static final PrintStream err
}
```

- out is reference of java.io.PrintStream class. Out is delcared is public static final field inside System class.
- Syntax : System.out
- print(), println() and printf are non static overloaded methods of java.io.PrintStream class.
    1. public void print(Object obj)
    2. public void print(String s)
    3. public void println(Object x)
    4. public void println(String x)
    5. public PrintStream printf(String format, Object... args);
- print() method print output on terminal but keep cursor on same line.
- println() method print output on terminal but moves cursor to the next line.
- If we want to print formated output on console/terminal then we should use printf() method.
- Java compiler generates .class file per Type(class/interface/enum) defined in .java file.
- If .java file do not contain Type then compiler do not generate .class file.

Java Language Features / Buzzwords

1. Simple
2. Object Oriented
3. Architecture Neutral
4. Portable
5. Robust
6. Multithreaded
7. Dynamic
8. Secure
9. High Performance
10. Distributed

Java is Simple Programming Language.

- Java is derived from / extended from C and C++.
- Syntax of java is simpler than C and C++
    1. No need to include header file.
    2. Java do not support structure and union
    3. Java do not support declaration and definition seperatly. Everything is definition
    4. Java do not support constructor's member initializer list and default argument
    5. Java do not support delete operator and destructor
    6. Java do not support copy constructor and assignment operator function.
    7. Java do not support operator overloading

8. Java do not support friend function and class.
9. Java do not support multiple class inheritance (multiple implementation inheritance) Hence there is no diamond problem and virtual base class.
10. Java do not support advanced typecasting operators
11. Java do not support pointer and pointer arithmetic.
12. Java do not support private and protected mode of inheritance
13. Java do not support sizeof operator
14. We can not write code globally.

- Since size of software which is required to develop java application is small, java is considered as simple.

Java is Object Oriented Programming Language.

- Since java supports all the major and minor pillars of oops, it is considered as object oriented.
- Major Pillars

1. Abstraction
2. Encapsulation
3. Modularity
4. Hierarchy

- Minor Pillars

1. Typing
2. Concurrency
3. Persistance

Java is architecture neutral programming language.

- CPU architecture : ARM, X86/64, POWER PC SPARC, ALPHA etc.
- java compiler is responsible for converting java source code into Bytecode.
- Bytecode is architecture neutral code hence java is considered as architecture neutral.

Java is portable programming language

- Java is portable because it is architecture neutral.
- By installing JVM, we an run same java application on any platform(OS) hence Java's slogan is "Write Once Run Anywhere" (WORA).
- Size of data types on all the platform is constant hence java is truely portable

1. boolean Not specified
2. byte 1 byte
3. char 2 bytes
4. short 2 bytes
5. int 4 bytes
6. long 8 bytes
7. float 4 bytes
8. double 8 bytes

- Since java is portable, it doesnt support sizeof operator.

Java is robust programming language

1. Since java is architecture neutral, it is robust
2. Since java is object oriented, it is robust
3. Since JVM automatically manages memory, java is robust
4. java is robust because of its exception handling.

Java is multithreaded programming language

- Program is execution / running instance of program is called process. Process is also called as task.
- Sub process / light weight process is called thread.
- thread is non java / os resource.
- An ability of operating system to execute single process at a time is called single tasking. e.g MS DOS
- An ability of operating system to execute multiple processes at a time is called multi tasking. e.g all modern os
- Multitasking can be achieved using

1. process
2. thread

- If any application use ony one thread then it is called single threaded application.
- If any application uses multiple thread then it is called multi threaded application.
- When starts execution of java application, it also starts execution of main thread and garbage collector. Because of these two threads, every java application is multithreaded.
- Main Thread
    1. It is User Thread / Non Daemon Thread
    2. It is responsible for invoking main method.
    3. It's default priority is 5.
- Garbage Collector
    1. It is also called as finalizer
    2. It is Daemon Thread / Background Thread
    3. It is responsible for deallocating / releasing / reclaiming memory of unused objects.
    4. It's default priority is 8.
- Thread is OS resource. To access os thread, java application developer need not to do os specific code. Rather SUN/ORACLE developer has already given thread framework. It means that java supports multi threading.

Java is dynamic programming language

- Since all the methods of class are by default virtual, java is considered as dynamic.
- We can run/execute java application on any upcoming hardware as well as operating system hence java is considered as dynamic programming language

Java is secure programming language

- To achive security, SUN/ORACLE has develped security framework i,e jav support security.
- java.security.* contain types related to security.

Java is high performance programming language

- Performance of java is slower than C and C++.
- JNI can help to access native code in java code which can improve performance.
- During execution, JIT compiler use optimization technique which improves performance of java application.

Java is distributed programming language

- Since java supports RMI, java is distributed programming language.

Data Type(s)

1. Primitive Data Types / Value Types
2. Non Primitive Data Types / Reference Types

Primitive Data Types

- Instance of primitive type get space on stack section.
- To create instance of primitive type new operator is not required.

1. boolean Not specified false
2. byte 1 byte 0
3. char 2 bytes \u0000
4. short 2 bytes 0
5. int 4 bytes 0
6. long 8 bytes 0L
7. float 4 bytes 0.0f
8. double 8 bytes 0.0d

- In java, primitive types are not classes. But for every primitive type class is given it is called wrapper class.
- Wrapper classes are defined in java.lang package.

1. boolean java.lang.Boolean
2. byte java.lang.Byte
3. char java.lang.Character
4. short java.lang.Short
5. int java.lang.Integer
6. long java.lang.Long
7. float java.lang.Float
8. double java.lang.Double

Non Primitive Data Types

- Instance of non primitive type get space on heap section.
- To create instance of non primitive type new operator is required.

1. Interface
2. Class

3. Enum
4. Array

- If we want to use any local variable then it is mandatory to store value inside it.

```
public static void main(String[] args)
{
    int number;
    System.out.println(number);//Error
}
```

- If we want to concat state of any java instance to the string then we should use + operator.

```
int number = 10;
System.out.println("Number : "+number);
```

```
System.out.println("Result : "+10+20);//1020
System.out.println("Result : "+(10+20));//30
```

Widening

- It is the process of converting state of instance of narrower type into wider type.

```
int num1 = 10;
//double num2 = ( double )num1; //Widening
double num2 = num1; //Widening
```

- In case of widening, explicit typecasting is optional.

Narrowing

- It is the process of converting state of instance of wider type into narrower type.

```
double num1 = 10.5;
int num2 = ( int )num1; //Narrowing
//int num2 = num1; //Narrowing : Error
```

- In case of narrowing explicit type casting is mandatory.
- If we want to convert String into numeric type then we should use parseXXX() method of Wrapper class

```

```

```
int num1 = Integer.parseInt("10");
float num2 = Float.parseFloat("10.1f");
double num3 = Double.parseDouble("20.2d");
```

Console IO

- Console is a class declared in java.io package.

```
 Console console = System.console();
System.out.print("Name  :   ");
String name = console.readLine();
System.out.print("Empid  :   ");
int empid = Integer.parseInt(console.readLine());
System.out.print("Salary  :   ");
float salary = Float.parseFloat( console.readLine());
```

Entry Point Method

- "main" is entry point method in java.
- Syntax

```
public static void main( String[] args )
{   }
```

- Invoking main method is a job of JVM.
- We can define main method per class but only one main method can be considered as entry point method.
- We can overload main method in java.

Access Modifier

1. private
2. package level private(or default)
3. protected
4. public

Path and Classpath

- Path is environment variable of OS platform which is used to locate java language tools.
- Syntax to set PATH: export PATH=/usr/bin/
- Check status of PATH echo $PATH
- Classpath is environment variable of Java Platform which is used to locate .class file / .jar file.
- Syntax to set CLASSPATH:
  export CLASSPATH=./bin/
- Check status of CLASSPATH echo $CLASSPATH

Comments

- If we want to maintain documentation of source code then we should use comments.
- Types of Comments:

1. //Single line comment
2. /* Multiline comment */
3. /** Java Documentation comment. */

- If we want to process java documentation comment then we should use javadoc tool.

Class

- Class is collection of fields and methods.
- Since Structre and behavior of instance depends on class, it is considered as template, model or blueprint for instance.
- Class represents group of instances which is having common structure and common behavior.
- By defining class we can achieve encapsulation.
- In java, class members are by default considered as package level private.

Instance

- In java, object is also called instance.
- An entity, which is having physical existance is called instance.
- An entity, which is having state, behavior and identity is called instance.
- Process of creating instance from class is called instantiation.
- By creating instance we can achive abstraction.

Fields

- In java, data member is called field.
- Class can contain static as well as non static field.
- Non static field is called instance variable. Instance variables are designed to access using instance.

```
class Test
{
    int number; //Instance variable
}
class Program
{
    public static void main(String[] args)
    {
        Test t = new Test( );
        t.number = 10;
        System.out.println(t.number);
    }
}
```

- Instance variable get space once per instance.

- Static data member is called class level variable. Class level variable is designed to access using class name.

```
class Test
{
    static int number; //Class level variable
}
class Program
{
    public static void main(String[] args)
    {
        Test.number = 10;
        System.out.println(Test.number);
    }
}
```

- Class level variable get space once per class.
- Class is non primitive type hence to create its instance it is mandatory to use new operator.
- If we want to perform operations on instance then it is mandatory to create reference / object reference of it.

```
Complex *ptr = new Complex(); //C++
//ptr is pointer
```

```
Complex c1 = new Complex(); //Java
//c1 is reference
```

```
Compex c1 = new Complex();
Complex c2 = c1;
Compex c3 = new Complex();
```

- Process of calling method on instance is called message passing
- this is implicit reference variable that is available in every non static method of the class which is used to store reference of current instance.