Day 9

Variable Arity Method

- A method which can accept variable number of arguments is called variable arity method / variable argument method.

```
public class Program {
    //Variable argument  / variable arity method.
    public static void sum( int... arguments ) {
         int result = 0;
         for( int element : arguments )
             result = result + element;
         System.out.println("Result :   "+result);
    }
    public static void main(String[] args) {
        Program.sum( );
        Program.sum( 10, 20, 30 );
        Program.sum( 10, 20, 30, 50, 60 );
        Program.sum( 10, 20, 30, 50, 60, 70, 80, 90, 100 );
    }
}
```

- Example:

    1. public PrintStream printf(String format, Object... args);
    2. public static String format(String format, Object... args);
    3. public Object invoke(Object obj, Object... args);

- Question : Write a program to print "Hello World" without giving semicolon.

```
public class Program {
    public static void main(String[] args) {
        if ( System.out.printf("Hello World") != null ) {
        }
    }
}
```

Java Comments

- If we want to maintain documentation of source code then we should use comment.
- Types of comments in Java
    1. //Single line comment
    2. /* Multiline comment */
    3. /** Java Documentation/Java Doc comment */
- Example

```
/**
     * Returns the {@code char} value at the
     * specified index. An index ranges from {@code 0} to
     * {@code length() - 1}. The first {@code char} value of the sequence
     * is at index {@code 0}, the next at index {@code 1},
     * and so on, as for array indexing.
     *
     * <p>If the {@code char} value specified by the index is a
     * <a href="Character.html#unicode">surrogate</a>, the surrogate
     * value is returned.
     *
     * @param      index   the index of the {@code char} value.
     * @return     the {@code char} value at the specified index of this
string.
     *             The first {@code char} value is at index {@code 0}.
     * @exception  IndexOutOfBoundsException  if the {@code index}
     *             argument is negative or not less than the length of
this
     *             string.
     */
public char charAt(int index) {
    if ((index < 0) || (index >= value.length)) {
        throw new StringIndexOutOfBoundsException(index);
    }
    return value[index];
}
```

Enum

- Consider code in C:

```
enum ArithmeticOperation{
    //Enum Constants/enumerator
    EXIT, SUM, SUB, MULTIPLICATION, DIVISION
};
enum ArithmeticOperation menu_list( void ){
    enum ArithmeticOperation choice;
    printf("0.Exit\n");
    printf("1.Sum\n");
    printf("2.Sub\n");
    printf("3.Multiplication\n");
    printf("4.Division\n");
    printf("Enter choice    :   ");
    scanf("%d", &choice);
    return choice;
}
int main( void ){
    enum ArithmeticOperation choice;
    while( ( choice = menu_list( ) ) != EXIT ){
        int result = 0;
```

```
        switch( choice ){
        case SUM:
            result = sum( 100, 20 );
            break;
        case SUB:
            result = sub( 100, 20 );
            break;
        case MULTIPLICATION:
            result = multiplication( 100, 20 );
            break;
        case DIVISION:
            result = division( 100, 20 );
            break;
        }
        printf("Result  :   %d\n", result);
    }
    return 0;
}
```

- If we want to improve readability of the source code then we should use enum.
- In Java, Using enum, we give name to any literal or group of literal.

```
enum Day{
    SUN("SunDay"), MON(2), TUES("TuesDay",3);
}
```

- Consider Example

```
enum Color{
    RED, GREEN, BLUE    //Name of enum constants
    //RED=0, GREEN=1, BLUE=2    //0,1,2 => Ordinal
}
```

- We can not change ordinal of enum constant.
- enum is a keyword in Java.
- enum is non primitive/reference type in Java.
- java.io.Serializable is empty interface. It is also called marker / tagging interface.
- java.lang.Comparable is a interface. "int compareTo(T other)" is a method of Comparable interface.
- java.lang.Object is a class. It is having 12 methods:
    1. toString()
    2. equals()
    3. hashCode()
    4. clone()
    5. finalize()
    6. getClass()
    7. 3 overloaded wait methods.

8. notify()
9. notifyAll( )

- java.lang.Enum is an abstract class which is considered as super class for enum.
- This class is introduced in JDK 1.5
- Methods of Enum class:
    1. public final Class getDeclaringClass()
    2. public final String name()
    3. public final int ordinal()
    4. public static <T extends Enum> T valueOf(Class enumType, String name);
- Consider enum in Java

```
enum Color{
    RED, GREEN, BLUE
}
```

- Compiler generate .class file per interface, class and enum.
- Compiler generated code for enum

```
final class Color extends Enum<Color> {
  public static final Color RED;
  public static final Color GREEN;
  public static final Color BLUE;

  public static Color[] values();
  public static Color valueOf( String str );
}
```

- If we define enum in Java, then it is implicitly considered as final class. Hence we can not extends enum.
- Enum constants are references of same enum which is considered as public static final field of the class. Please consider above code.
- values() and valueOf() are methods of enum which gets added at compile time.
- In Java, if we want to assign name to the literals then it is mandatory to define constructor inside enum. And to get value of the literal it is nessary to define getter methods inside enum.

```
enum Day{
    SUN("SunDay"), MON(2), TUES("TuesDay", 3);//Must be first line
    private String dayName;
    private int dayNumber;
    private Day(String dayName) {
        this.dayName = dayName;
    }
    private Day(int dayNumber) {
        this.dayNumber = dayNumber;
    }
    private Day(String dayName, int dayNumber) {
```

```
        this.dayName = dayName;
        this.dayNumber = dayNumber;
    }
    public String getDayName() {
        return dayName;
    }
    public int getDayNumber() {
        return dayNumber;
    }
}
```

Object Oriented Programming Structure/System(OOPS)

```
- It is not a syntax.
- It is a process/methodology that we can implement using any OO
programming language.
```

Major Pillars Of OOps

```
1. Abstrasction
2. Encapsulation
3. Modularity
4. Hierarchy
```

Minor Pillars Of OOps

```
1. Typing / Polymorphism
2. Concurrency
3. Persistence
```

Abstraction

- It is a major pillar of OOPS.
- It is a process of getting essential things from system.
- Abstraction focuses on outer behavior of instance.
- Abstraction focuses on the essential characteristics of some object, relative to the perspective of the viewer.
- Using abstraction, we can achieve simplicity.
- Abstraction in Java

```
Complex c1 = new Complex( );
c1.acceptRecord( );
```

```
c1.printRecord( );
```

- Creating instance and calling method on it is abstrasction in Java.

```
import java.util.Scanner;
Scanner sc = new Scanner( System.in );
int number = sc.nextInt( );
```

Encapsulation

- It is a major pillar of oops.
- Defintion:
    1. Binding of data and code together is called encapsulation.
    2. To achive abstrasction, we need to implement some business logic. It is called encapsulation.
- Class implementation represents encapasulation.
- Encapsulation in Java

```
class Complex{
    //Fields of the class   => Data
    private int real;
    private int imag;
    //Methods of a class    =>  Code
    public Complex( ){
    }
    public void acceptRecord( ){
    }
    public void printRecord( ){
    }
}
```

- Encapsulation represents internal behavior of the instance.
- Abstraction and encapsulation are complementary concepts: Abstraction focuses on the observable behavior of an object, whereas encapsulation focuses on the implementation that gives rise to this behavior.
- Proces of declaraing field of the class private is called hiding. Hiding represents data encapsulation.

```
class Employee{
    private float salary;   //Hiding
    public void setSalary( float salary ){
        if( salary > 0 )    //Data Security
            this.salary = salary;
        else
            throw new IllegalArgumentException("Invalid Salary");
    }
}
```

- Data hiding helps to achieve data security.
- Process of giving controlled access to the data is called data security.

Modularity

- It is a major pillar of oops.
- Process of developing complex system using small parts/modules is called modularity.
- If we want to minimize module dependancy then we should use modularity.
- With the help .jar file, we can achive modularity.

Hierarchy

- It is a major pillar of oops.
- A level/order/ranking of abstraction is called hierarchy.
- Using hierarchy, we can achive reusability.
- Advnatages of reusability:
    1. We can reduce development time.
    2. We can reduce development cost.
    3. We can reduce develoeprs effort.
- Types of Hierarchy:
    1. Has-a/part-of => Association
    2. Is-a/kind-of => Inheritance/Generalization
    3. Use-a => Dependancy
    4. Creates-a => Instantiation

Typing / Polymorphism

- It is minor pillar of oops.
- An ability of instance to take multiple forms is called polymorphism.
- Polymorphism = Poly(many) + morphism(forms/behavior)
- Types of polymorphism:
    1. Compile time polymorphism
        - Method Overlaoding
    2. Runtime polymorphism
        - Method Overriding
- Using polymorphism, we can reduce maintenance of the system.

Concurrency

- It is minor pillar of oops.
- An ability of OS to execute multiple processes simultaneously is called Concurrency.
- In context of oops it is called concurrency.
- If we want to utilize H/W resources(CPU) efficiently then we should use concurrency.
- In Java, using thread, we can achive concurrency.

Persistance

- It is minor pillar of oops.
- Process of storing state of instanec on HDD is called Persistance.
- Using persistence, we can maintain state of instance on secondry storage.
- Using File IO and JDBC, we can implement persistence.

System Date

```
import java.time.LocalDate;

LocalDate ldt = LocalDate.now();
int day = ldt.getDayOfMonth();
int month = ldt.getMonthValue();
int year = ldt.getYear();
```