Core Java

Q: What is computer program ?

Ans. It is simply a list of instructions that tell a computer what to do.

```
package ------;
public class Hello {
    public static void main (String [] args) {
        System.out.println ("Hello there ");
    }
}
```

Hello there

★ Variable =>

It is a place where we can save the data.

★ Data type =>

It specifies what kind of data goes into something.

```
public class Variables {
    public static void main (String [] args ) {
        int x;
        x = 34;
        System.out.println (x);
    }
}
```

34

```
public class VariablesContinued {
    public static void main(String[] args) {
        int var = 100000;
        long Number = 1000000000L;
        short num = 32768;
        byte reallySmall = 127;          (-128 to +127)
        double decimalVar = 394.003;
        boolean decision = true/false;
        char letter = 'g';               (single char only)
        string words = 'A B C D';
    }
}
```

☆ Arrays =>

These are used to store multiple related things in one variable

```
public class LearningArrays {
    public static void main(String[] args) {
        int[] values = new int[100];
        values[0] = 1000;
        values[99] = 234567;
        System.out.println(values[0]);
        String[] words = new String[] {"A", "B", "C"};
        System.out.println(words[0]);
    }
}
```

Haq, ek behtar zindagi ka.

Saturday, October 21

★ Control flow using if-else and switch statements

```
public class ControlFlow {
    public static void main(String[] args){
        boolean hungry = true;
        if(!hungry){
            System.out.println("Im starving");
        }else{
            System.out.println("Im not hungry");
        }
    }
}
```

Im not hungry

```
int month = 6;
String monthString;
switch(Month){
case 1: monthString = "January";
  break;
case 2: monthString = "February";
  break;
default: monthString = "unknown Month";
  break;
}
System.out.println(MonthString);
```

Sunday, October 22

Unknown Month

Haq, ek behtar zi[illegible]gi ka.

2022/03/13 10:42

Methods

- It is a collection of statements that perform some specific task and return the result of the caller.
- It can perform some specific task without returning anything.
- It allows us to reuse the code without retyping the code.

```
public class LearningMethods {
    public static void main (String [] args ) {
        System.out.println ("Abcdefgh");
    }
    printSomeJunk ("My name");
    printSomeJunk (34);
}
public static void printSomeJunk (String arg) {
    System.out.println ("Some bla bla " + arg );
}
public static void printSomeJunk (int arg) {
    System.out.println ("Integer passed in: " + arg );
}
```

Method — Argument

Method — Parameter

Method signature

Abcdefgh
Some bla bla My name
Integer passed in: 34

[can allowed in diff. class

, ek behtar zindagi ka.

```
public class MyUtils {
public static void sum2Num (int firstArg, int secondArg){
    System.out.println (firstArg + secondArg );
}
```

int → Return type.

```
public static addTo (int someArg) {
    int result = someArg + 10;
    return result;
```

(return type not available in void method
change return type)

```
}
}
```

Main in Class

```
int var = MyUtils.addTo(67)
System.out.println(var);
```

77

Static vs Instance variable →

```
Public static void/int ____ ( ) {
Class name. Method name ( );
}
```

(static)

```
Public void/int ____ ( ) {
class name variable;
variable = new class name ( );
variable. method name;
}
```

(Instance / non-static)

Haq, ek behtar zindagi ka.

Wednesday, October 25

★ Class & objects →

- An object is an instance of a class. It is a template or blueprint from which objects are created. So, an object is an instance (result) of class.
  - It is real-world entity.
  - It is run time entity.
  - Object is an entity which has state & behaviour.

- Class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.
  - A class can contain -
    Fields,
    Methods,
    Constructors,
    Blocks,
    Nested class and interface.

- Name of constructor is same as that of a class.

```
public Human() {
}
```

Constructor.

```
Human tom;
tom = new Human();
```

object creation

9, ek behtar zindagi ka.

2022/03/13 10:42

Thursday, October 26

Class Animal

```
public class Animal {
    int age;
    String gender;
    int weightinLbs;
    public Animal(int age, String gender, int weightinLbs){
        this.age = age;
        this.gender = gender;
        this.weightinLbs = weightinLbs;
    }
    public void eat() {
        System.out.println("Eating...");
    }
    public void sleep() {
        System.out.println("sleeping...");
    }
}
```

Class Zoo

```
public class Zoo {
    public static void main(String[] args) {
        Animal animal1 = new Animal(12, "M", 23);
        animal1.eat();
        animal1.sleep();
    }
}
```

Eating

sleeping

Friday, October 27

★ Instance variable →

- It is used to maintain state for that particular object.

★ Class inheritance →

- It is an other way of organizing our code.
- One class inherits behaviour from another class.

★ Interface →

- It is a blueprint of a class. It has static constants and abstract methods.
- It is a mechanism to achieve abstraction. There can be only abstract methods in the java interface, not method body.

Syntax -

```
interface <interface_name> {
// declare const. fields
// declare methods that ~~is~~ abstract
// by default
}
```

★ Abstract classes →

It is almost a regular class. A class which is declared with the abstract keyword.

ek behtar zindagi ka.

Saturday, October 28

- It can have abstract & non-abstract Method (Method with the body).
- It needs to be extended and its method implemented.
- It can't be instantiated.

Syntax :-

```
abstract class Bike {          → Abstract class.
    abstract void run();       → Abstract Method.
}
class Honda4 extends Bike {
    void run() {
        System.out.println("running safely");
    }
    public static void main(String args[]) {
        Bike obj = new Honda4();
        obj.run();
    }
}
```

19-2-22

★ Working with strings →

Sunday, October 29

- length
- substring (1)
- equals
- charAt(1)
- indexOf("...")
- substring(1,2)
- indexOf("...", 1) — start from here
- lastIndexOf("...")

Haq, ek behtar zindagi ka.

Monday, October 30

★ WHILE loops →

★ FOR loops →

★ Nested FOR loops →

★ JAR file → Java ARchive

★ Exception Handling →

- It is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.
- The core advantage of it is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application: i.e. why we need to handle exceptions.

★ TRY → Defines the code block to run.

★ CATCH → Defines a code block to handle any error.

★ FINALLY → Defines a code block to run regardless of the result.

★ THROW → Defines a custom error.

The catch and finally are optional, but we must use one of them.

, ek behtar zindagi ka.

2022/03/13 10:43

★ Collections →

- It is a framework that provides an architecture to store and manipulate the group of objects.
- Java collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation and deletion.
- It means a single unit of objects. It provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

★ Collection Methods →

- add()
- remove()
- contains()
- clear()
- isEmpty()
- retainAll()

★ Generics →

It means parameterized types. The idea is to allow type (Int, string, ---etc and user-defined types) to be a parameter to methods, classes and interfaces. Using generics, it is possible to create classes that work with different data types. An entity such as class, interface or method that operates on a parameterized type is a generic entity.

Haq, ek behtar zindagi ka.

Wednesday, November 01

/Generic type.

- Generic Class →

Like C++, we use <> to specify parameter types in generic class creation. To create objects of a generic class, we use the following syntax:

```
// To create an instance of generic class
~~Base type~~
BaseType <Type> obj = new BaseType <Type>();
```

Note:- In parameter type we cannot use primitives like 'int', 'char' or 'double'.

A generic class is implemented exactly like a non-generic class. The only diff. is that it contains a type parameter section. There can be more than one type of parameter, separated by a comma. The classes, which accept one or more parameters, are known as parameterized classes.

★ Generic Method →

Generic java method takes a parameter and returns some value after performing a task. It is exactly like a normal function, however, a generic

method has type parameters that are cited by actual type. This allows the generic method to be used in a more general way. The compiler takes care of the type of safety which enables programmers to code easily since they do not have to perform long, individual type castings.

* Generic functions →

We can also write generic functions that can be called with different types of arguments based on the type of arguments passed to the generic method. The compiler handles each method.

- Type parameters in java Generics -

The type parameters naming conventions are important to learn generics thoroughly. The common type parameters are as follows -:

- T - type
- E - Element
- K - Key
- N - Number
- V - Value

- Advantages -

- Code reuse
- Type safety

Haq, ek behtar zindagi ka.

Friday, November 03

★ Generics with wildcards →

It is a mechanism in java generics aimed at making it possible to cast a collection of a certain class, e.g:, A to a collection of a subclass or superclass of A.

- Generic wildcards -

- They target 2 primary needs.
  - Reading from a generic collection.
  - Inserting into a generic collection.

There are 3 ways to define a collection (variable) using generic wildcards.
These are :-

1. `List<?> listOfUnknown = new ArrayList<A>();`
2. `List<? extends A> listOfUnknown = new ArrayList<A>();`
3. `List<? super A> listOfUnknown = new ArrayList<A>();`

★ Multithreading →

It is a java feature that allows concurrent execution of 2 or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Haq, ek behtar zindagi ka.

22-2-22 Saturday, November 04

Threads can be created by using two mechanisms -

① Extending the thread class -
We create a class that extends the java.lang.thread class. This class overrides the run() method available in the thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. start() invokes the run() Method on the thread object.

② Implementing the runnable interface -
We create a new class which implements java.lang.Runnable interface and override run() method. Then we instantiate a thread object and call start() Method on this object.

★ JDBC → (Java Database Connectivity)
It is a java API to connect and execute the query with the database. It is a part of JavaSE (Java std. Edition). JDBC API uses JDBC drivers to connect Sunday, November 05
with the database. There are 4 types of JDBC drivers -:

- JDBC-ODBC Bridge Driver.
- Native Driver.
- Network Protocol Driver.
- Thin Driver.

Ha ek behtar zindagi ka.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like open Database connectivity (ODBC) provided by Microsoft.

★ JShell →

It is an interactive Java shell tool, it allows us to execute java code from the shell and shows output immediately. JShell is a REPL (Read Evaluate Print Loop) tool and run from the command line.

- Advantages -

JShell has reduced all the efforts that have required to run a java program and test a business logic.

★ Lambda Expression →

It is a new and imp. feature of java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.

Haq, ek behtar zindagi ka.

It is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.

It is treated as a function, so compiler does not create .class file.

Syntax:-

(argument-list) -> {body}

It is consisted of 3 components.

1. Argument-list → It can be empty or non-empty as well.
2. Arrow-token → It is used to link arguments-list and body of expression.
3. Body → It contains expressions and statements for lambda expression.

★ Functional interface →

Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface. Java provides an annotation @FunctionalInterface, which is used to declare an interface as functional interface.

Haq, ek behtar zindagi ka.

★ Streams →

Introduced in java 8, the stream API is used to process collections of objects. A stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.
The features of java stream are -

- It is not a DS instead it takes I/P from the collections, Arrays or I/O channels.
- Streams doesn't change the original DS, they only provide the result as per the pipelined methods.
- Each intermediate operation is lazily executed and returns a stream as a result, hence various intermediate operations can be pipelined. Terminal operations mark the end of the stream and return the result.

Different operations on streams -

Intermediate Operations:

1. Map
2. Filter
3. Sorted

Terminal operations:

1. Collect
2. ForEach
3. Reduce

Haq, ek behtar zindagi ka.

2022/03/13 10:44