Catnues Learning Framework - Java

20/07/2020

Java Basics:

→ JRE vs JDK

object oriented language.

Java → Programming language (Syntax, Data types, Control Flow, object oriented)

→ Runtime Environment (Configuration, Security, threading, Input/output)

↓

JavaSE, JavaEE,
JavaME, JavaFX

JRE → run Java Apps

JDK → Create Java apps

Integrated Development Environment (IDE)

Comments:

→ Line Comments → //

→ Block Comments → /* .... */

→ JavaDoc Comments → /** .... */

Packages:-

→ Package Provide organization, Affect Source Code File Structure.

Operator:

Postfix → x++, x--

Prefix → ++x, --x

multiplicative → * / %

Additive → + -

Type Conversions:

→ Implicit type Conversion → Performed automatically by the Compiler

int ival = 50;
→ long lval = ival;

→ Explicit type conversion → Explicitly in code with cast operator.

→ long lval = 50;
int ival = (int) lval.

Relational Operators:-

-> >	Greater than
-> >=	Greater than or equal to
-> <	less than
-> <=	less than or equal to
-> ==	equal to
-> !=	not equal to

Logical operators:-

-> And - & -> true & true

-> Or - | -> true|false, true|true

-> Exclusive or (XOR) — ^ -> f^t, t^f

-> Negation - ! -> !not false is ~~tr~~ true

->Conditional:-

-> Condition and -> && - true && true

-> Conditional Or -> || - False||true, true||.

-> Conditional Assigment

-> If-else Statement

```
if(condition)
    true Statement;
else
    False -Statement;
```

```
if (condition-1)
    Statement 1;
else if (condition-2)
    statement -2;
    :
else if (condition -N)
    Statement N;
else
    False Statement;
```

~~For each loop~~:-

-> ~~For (loop variable declaration~~

-> Block Statements

```
{
  Statement -1;
  Statement-2;
     :
  Statement -N;
}
```

-> loops:

-> while loop -> while (condition) statement;

-> do-while loop -> do{statement} while (condition)

-> for loop -> for(initialize; condition, update)

Statement;

Arrays:- Provides an ordered collection of elements, Each element accessed via an index. index ~~Sta~~ range from 0 to number of elements minus 1

For-each loop:- Statement once for each members in an array

```
-> for (loop-variable -declaration : array)
       Statement ;
```

Ex:

```
for (Float CurrentVal : the vals)
    Sum += CurrentVal ;
```

Switch:- Transfer control to a statement based on a value.

```
-> switch(test-value){
       Case value -1:
            Statements
       Case value -2:
            Statements
       :
       Case value -n:
            Statements
       default:
            Statements
   }
```

```
int ival = 10;
switch (ival % 2){
   Case 0:
      System.out.println(ival);
      "      "      " ("is even");
      break;
   Case 1:
      System.out.Print(ival);
      "      "      ("is odd");
      break;
   default:
      System.out.println("oops it broke");
      break;
}
```

Classes: Provide astructure for describing and creating objects

Class classname{
}

Encapsulation:- internal representation of an object is generally hidden
Uses access modifies to achieve encapsulation.

access modifiers:
- → Public Everywhere
- → Private → only within its own class
- → Protected → only within its own class and SubClasses

Special References:- this and null

this:- is an implicit reference to the Current Object, reducing ambiguity

null:- is a referen6 literal, Represents an uncreated object, Can be assigned to any reference variable.

Accessors and Mutators: uses to Control field access.

→ Accessors retrieves field values, Also Called getter

method name: getFieldName

→ mutators modifies Field value, Also Called Setter

method name: set FieldName

Mechanisms For Establishing Initial State:-

→ Java Provides 3 mechanisms

- → Field initializers → ~~is part of object Construction Fields receive a "zero" value by default~~ → Allows to specify a Field's initial value as a part of declaration
- → Constructors → Executable Code used during object creation to set the initial state
- → Initialization blocks → Shared across all Constructors

Parameters immutability:- → Passed by making a copy of the value. Known as [illegible] Passing 'by-value'

→ Changes made to passed value are not visible outside of method

→ Changes made to members of passed class instances are visible outside of method

overloading:- A class may have multiple versions of its constructors or methods. Each constructor and method must have a unique signature and signature is made up of 3 parts, 1) Numbers of parameters 2) Type of each parameter 3) Name

Inheritance ~~&~~ Constructors:-

Class inheritance:- A class can be declared to inherit from another class using "extends" keyword
Desired class has characteristics of base class and add it's own specifications.
→ Can be assigned to base class typed references
→ Field hide base class fields with same name
→ methods override base ~~class~~ class methods with same signature

Object class:- is the root of the java class hierarchy (Every class has the characteristic of the object class

→ Every class inherits directly or indirectly from the object class.

methods:- clone, hash code, getclass, finalize, tostring, equals

~~[illegible]~~ :- Special Reference :- Super. → Similar to 'this', Super is an implicit reference to the Current object

Controlling inheritance and overriding :- By default all Classes can be extended and derived Classes have the option to use or override inherited methods
→ ~~It has~~ A Class can Change these default
↳ use final to prevent inherit/override
→ use abstract to require to inherit/override

→ Constructors are not inherited
→ A base Class Constructor must always be Called
↳ By default, base Class no-argument Constructor is Called
↳ Can explicitly call a base class Constructor using Super followed by Parameter list
must be First line of Constructor.

String class :- It Stores a sequence of unicode Characters
↳ Literals are enclosed in double quotes(" ")
↳ Values can be concatenated using + and +=
↳ String objects are immutable

methods :- Length
Value of
Concate, replace, toLowercase, toUppercase, trim, split
Format
CharAt, Substring, Contains, endsWith, StartsWith, indexOf
CharAt, Substring, Contains, endsWith, StartsWith, indexof
lastindexOf, CompareTo, CompareToIgnoreCase, isEmpty, equals,
equals IgnoreCase

Converting non-String to String:-

- String.valueOf provides overrides to handle most types
- conversions often happen implicitly
- class conversion controlled by the class' toString method

String Builder:-

StringBuilder provides mutable String Buffer for best performance
Pre-size buffer; methods:- append and insert

Classes:-
-> Common interaction through object class
-> Fields and methods specific to the type
-> Incurs an overhead cost

Primitives:-
-> Lightweight
-> Cannot expose fields or methods
-> Cannot be treated as Object

Primitive Wrapper class hierarchy:-

$$\begin{tikzcd}
 & \text{object} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Boolean} & \text{Number} & \text{Character}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Byte} \arrow[r, dash] & \text{Short} \arrow[r, dash] & \text{Integer} \arrow[r, dash] & \text{Long} \arrow[r, "\text{Number}"] & \text{Float} \arrow[r, dash] & \text{Double}
\end{tikzcd}$$

Primitive wrapper classes:
-> Hold primitive values
-> Capabilities and overhead of Classes
-> All class instances are immutable

primitive to wrapper -> boxing.
wrapper to primitive -> unboxing.

String to primitive (parseX)
String to wrapper (valueOf)

Using wrapper classes :- -> Treat as Object
-> Null references

Range.

int	-128 to 12	
Byte	-128 to 127	
Short	-128 to 127	
Char	'\u0000' to	'\uffff'
Boolean	True or False	

Class.

Byte
Short
Integer
Long
MIN_VALUE, MAX_VALUE, bitCount
toBinaryString

Float
Double -> MIN_VALUE, MAX_VALUE
isInfinite, isNaN

Character -> MIN_VALUE, MAX_VALUE
isDigit, isLetter

Boolean -> TRUE, FALSE

Final Fields:- ->It prevents from being changed once assigned, It must be set during creation of an object instance (Field initializer, initialization block, or Constructor)

->Adding the static modifier makes a final field a named Constant cannot be set by an object instance.

Enumeration:-

-> useful for defining a type with a finite list of valid values

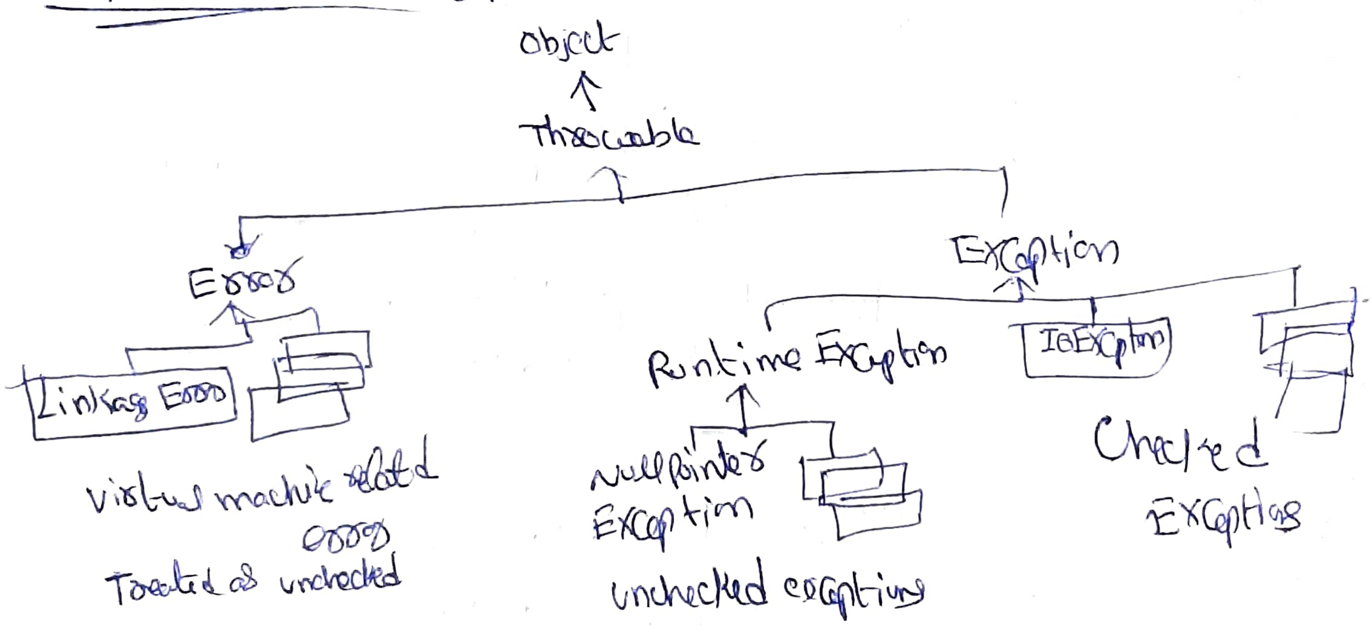
-> enum Keyword

-> provide a Comma-Separated value list

Error Handling with Exceptions:- needs to be implicit in application developer

Exceptions provide a non-intrusive way to signal errors (try/catch/finally)

Exception class hierarchy:-



Package:- is a group of related types
->create namespace
-> provide an access boundary
->Act as a unit of distribution

Type import:- Single type import -> Preferred way
~~Import~~ Import on demand -> use with Caution

Interface:- defines a Contract, Class implement interfaces, interfaces don't limit other aspects of the Class implementation.
-> Some interface requires addition type information called as generics
-> Classes are free to implement multiple interfaces

Declaring an interface:

methods:	Constants	Extending interfaces:
Name, Parameters, and return type	Typed name values	-> An interface can extend another interface
Implicitly public	Implicity public, final, Static,	-> extended interface implies implementation of base

Static Members: are Shared Class-wide
Declare using the static Keyword

- Field
  - -> A value not associated with a specific instance
  - -> All instances access the Same value
- method
  - -> Performs an action not tied to a specific instance
  - -> Can access static fields only.

Static import:- Short hand for accessing Static members

static initilization Blocks: Performs one-time type initialization, precede with static keyword.

~~Nested~~ Nested types:- is a type declared within another type
- -> Class can be declared within Classes and interfaces
- -> Interfaces can be declared within Classes and interfaces

-> It Supports all members of access modifiers (Public, private, Protected, Package Private)

→ structure and scoping → static classes nested within classes
→ All classes nested within interfaces
→ All nested interfaces

<u>Inner class</u>:- → Each instance of the nested class is associated with an instanc
of the enclosing class
→ Non static classes nested within class

<u>Anonymous classes</u>:- are declared as part of their creation, Anonymous
classes are inner classes