27/07/2022

Java Notes Day5

Java documentation - https://docs.oracle.com/javase/8/docs/api/index.html

Day 5
1. Late binding and Early binding
2. Interfaces
3. Packages
4. Access specifiers
5. Exception handling

Binding depends upon the overriding procedure
Parent method - in the super class is overriden
child hides the method of the parent class while overriding

Late binding happens at run time

Example - Lets say we have class A {}
class B extends A {}
Now creating two references, a1(obj of class A), b1(obj of class B)
We have method void show() in class A - This can be called using a1 or b1
If we have the same class overridden in class B, then b1 calls the method of class B only
If we create another class A obj ref as a2 and
Assign it with value of b1 ref - a2 = b1;
Now call the a2.show() - The compiler shows method of class A
But now, this decision is changed during the time of execution and the overriden method of class B is run

Binding occurs at reference.methodCall()
Early binding occurs before Late binding

Reference of child class can point to parent class method

Late binding is only possible for inherited or overridden methods*****

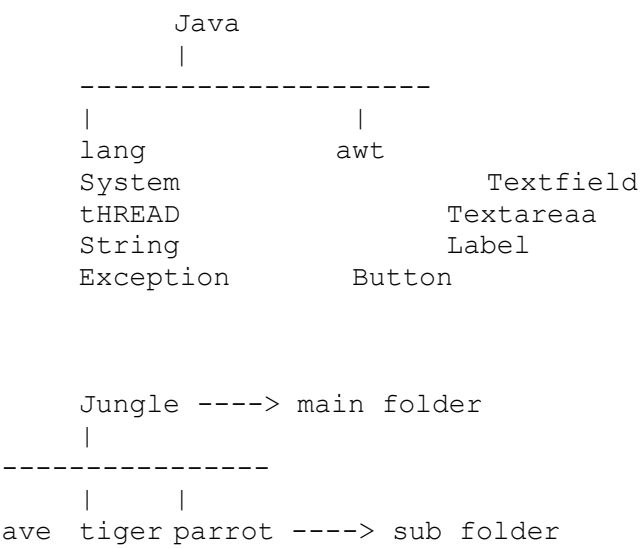
Interface
-Pure abstract class
-Overriding is not mandatory for children
-Implementation is cumpulsory for the implementing class

Lets say there is an interface X{void photo();}
class Photographer implements X
Then the reference of object of Photographer can be used instead of X reference

throw new RuntimeException("Insufficient funds")

abstract class can inherit from a player

Package
- similar to a directory
- store all the relevant files



Access specifiers in Java

private
>private can only be reffered by the declaring class

public
>Can be reffered by anyone from anywhere

protected
>Can be only be referred by the child class via extends keyword i
any package
>Can be referred by the non-child class within the same package

default - no access specifier
>Can only be referred by the same package classes, child/non-child
class

Errors
> Compile time error - syntax
>> Runtime - exceptions
> Linker - library
> Fatal - JVM
> Logical - unexpected output

Exception - Synonymous with runtime error

Keywords - throws, finally, try, catch

->Object

```
-> Throwable
      -> Error - Error cannot be caught
      -> Exception - Can be try and caught(checked)
            -> RuntimeException(unchecked exceptions) - Superclass
                  -> ArithmeticException - eg - divide by zero
```

checked and unchecked - by the compiler
OutOfBounds - exception thrown when we try to access a member of data that is out of bounds

```
void fundTransfer(sa, ta, amt)
      1. check presence of the target(ta)
            if ta found           else
            then                  ta not found
      2. check presence of the source(sa)
            if sa found           else
            then                  sa not found
      3. check balance at sa
            if sa has sufficient balanceelse
            then                  sa insufficient balance
      4. sa.withdraw()
      5. ta.deposit()


void fundTransfer(sa, ta, amt)

try
   {
      1. check presence of the target(ta)
      2. check presence of the source(sa)
      3. check balance at sa
      4. sa.withdraw()
      5. ta.deposit()
    }

catch(if ta not found)
  {
      ta not found
  }
catch(if sa not found)
  {
      sa not found
  }
catch(if sa with insufficient balance)
  {
      sa has insufficient balance
  }

JVM

      if(j is non zero)
      cpu
      else
      ArithmeticException e = new Arithmeticexception();
      throw e;
```

InputMismatchException - java.util
StringIndexOutOfBounds

Unhandled exception - An exception that is expected but not declared

Checked   - Automatically thrown by the JVM
Unchecked - Must be thrown by the user

Custom Exception
      > Defined by the user as a child of the Exception class
      > Must be defined as throws in function call, and also must be caught(try catch block) in main method