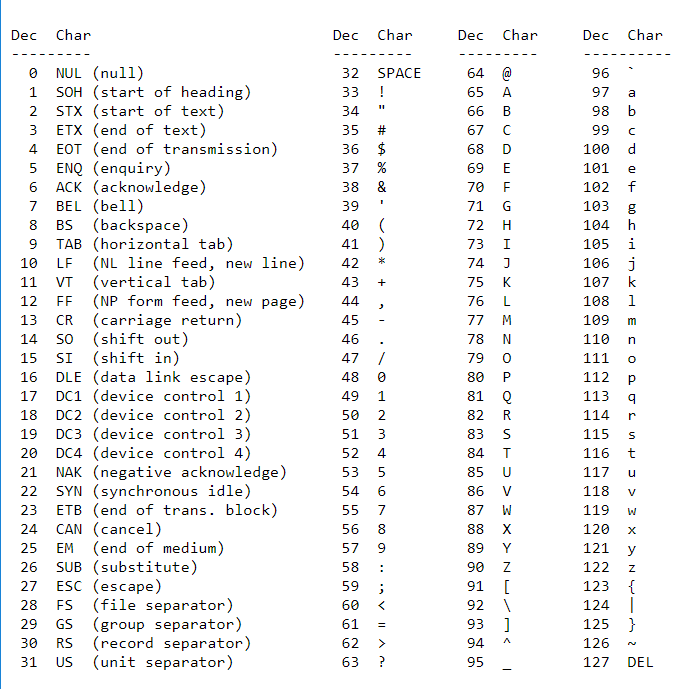
Src ( source where you keep ur java code source)  
Java is a OOP(object oriented programming focus) while python is a both object oriented and procedural.  
  
**Variables.**  
Int -2147483648, +2147483647  
byte -128,127  
short 32767,  
long   
  
float 2.2222f  
double 2.22222222d

Java will automatically change to int.

For Char (int to ‘c’) ACSII table



**Methods**  
public static void method( inputs); { /\*\* stuff \*/}  
<accessibility optional returntype method(inputs)>  
   
eg. public static void calculateScore(Boolean gameOver, int score, int bonus){ /\*\* stuff \*/}  
 private static int calculateScore(Boolean gameOver, int score, int bonus){ /\*\* stuff \*/ return 1;}  
  
.toLowerCase())  
  
**Method Overloading**  
using same method with different number of input (thus multiple method with same name  
 **Control Flow Statement**  
**-If else**  
**-switch(value){** case 1:  
 break;  
 case 2:   
 break;  
 case 3: case 4: case 5:  
 break;  
 default:   
 break;  
}

**-for( init; termination; increment){}**eg. for(int x=0;x<10;x++;){}  
eg. int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
 for (int item : numbers) {….} //will loop through the list and make item each element of the list.

**-the while(count!=5){ increment }**while(x<3) {x++;}orwhile(true){   
 if(count==6){break;}  
 count ++;  
 }

**-Do{ count++;}while(count !=6) //this will always run once ,careful of infinite loop**

**Iterator // can access a collection class through and iterator.**<https://www.tutorialspoint.com/java/java_using_iterator.htm>Iterator itr = arrayList.iterator(); //Make an arrayList into iterator()  
while(itr.hasNext()) { // .hasNext() check if there is still elements, returns a Boolean  
String element = itr.next(); //.next() returns the next element.  
itr.remove();}//.remove() all elements in arrayList **Classes  
-public class Car{** private int doors;  
 private String model; // << this are known as **fields**. Variables of class  
   
 public void setModel(String model){ //Setter similar to a **constructor** Car(){} **same name as class**  
 this.model = model; } // a **method** access for private fields.   
 //&&“this.model” refers field private String model.

public String getModel( ) { //Getter  
 return this.model; // “this.model” refers variable listed in Car  
}-public class Main{ // **Main will always run first**…  
 public static void main(String[] args) {  
 Car porsche = new Car(); // “Car porsche” define a type and “new Car()” creates object  
 Car holden= new Car(); // Creating objects car  
 }  
 }  
 **Encapsulation -private  
when you changed a variable to private access // normally put setter/constructor and getter.   
only allow limited variables access, allowing validation..  
  
Inheritance  
A Basic class to allow other classes to copy its fields/methods. Eg. Dogs inherits from Animal class.  
-public class Animal {  
 private String name;  
 private int weight; }  
-public class Dog extends Animal{  
 private int tail; // only applicable to dog  
 public Dog( String name; int weight){ // generate the constructor of base class, (Animals)  
 // putting int tail; allows programmer to initialize.  
 super(name,weight); //Change to super(1,weight) means assuming weight Is present  
 @Overide // overriding methods of base class  
 super.eat(); //access base class method even when overwritten**

**Compare String**

**.equals**

<https://stackoverflow.com/questions/2452166/easiest-way-of-checking-if-a-string-consists-of-unique-characters><http://javahungry.blogspot.com/2014/11/string-has-all-unique-characters-java-example.html>

**Arrays**  
int[] myIntArray= new int[10];  
myIntArray[0]=45;

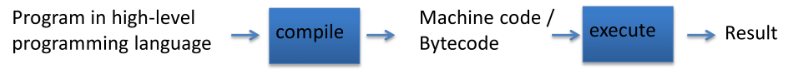
**List**  
**-ArrayList**  
private ArrayList<String> groceryList = new ArrayList<String>();  
groceryList.size();  
groceryList.set(1,”apple”);  
groceryList.get(1); //apple  
groceryList.remove(1);  
groceryList.clear(); //remove all  
groceryList.contain(“apple”); //return true or false ,Boolean  
groceryList.indexOf(“apple”); //return position of search item  
NewArrayList.addAll(1,2,2,2)//Fast way of duplclicating an arrayList if same array size  
Collections.addAll(groceryList,1,2,3,4,5); //duplicating without array size limit

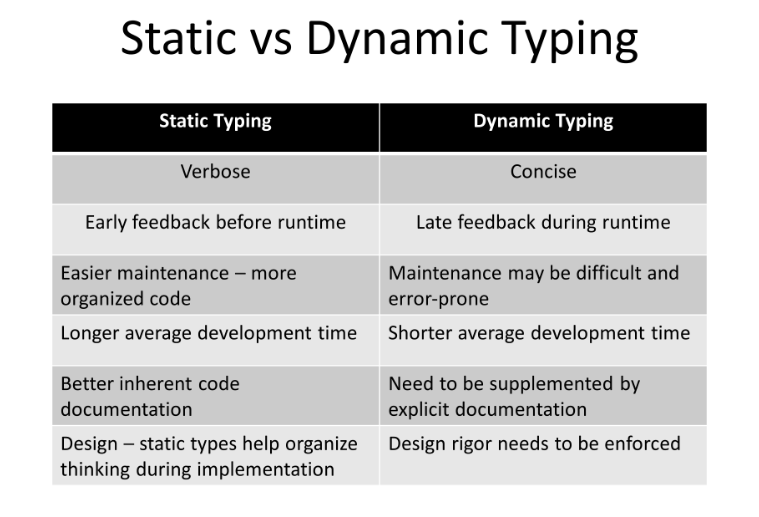
**Static vs Instance variable and method.**  
Static refers to stuff being shared across all objects of a class. (Eg. Bank acc: Interest rates )//static int   
Instance refers to stuff unqiue to the Object. (Eg. Bank acc: Name )

Static int ( creates one memory slot ) , an Instance int (have memoryslot for each object)

**Private**  
-Accessor Methods & Mutator Method (Getter and Setter)  
private int accNumber;   
public int getAccNumber(){ //Getter or Accessor method  
 return accNumber;  
 }

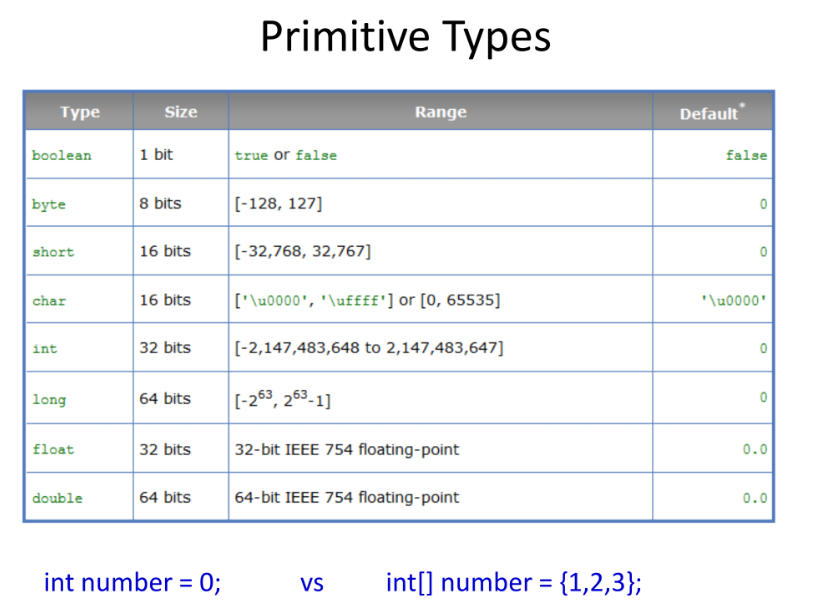
Public void setAccNumber(int accNumber){ //Setter or Mutator method  
 this.accNumber=accNumber;  
 }  
  
Class: Blue print to make objects  
-Const

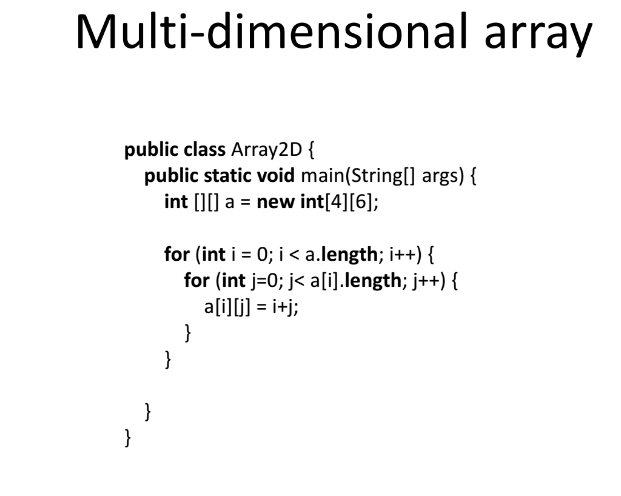
ructor  
-Methods  
  
Creat and object using classes  
  
  
  
**Static Typing**  
Java is a statically-typed language. (Error Appears at compilation phase.  
Python is not statically-typed. (No declaration of variable typed , Only see error at execution)  
  
  
  
**Automatic Checking provided by the programming language**  
Static Checking: Bug is found Before the program runs  
Dynamic checking: Bug is found automatically when the code is executed  
No Automatic Checking : The language does not help you find such error

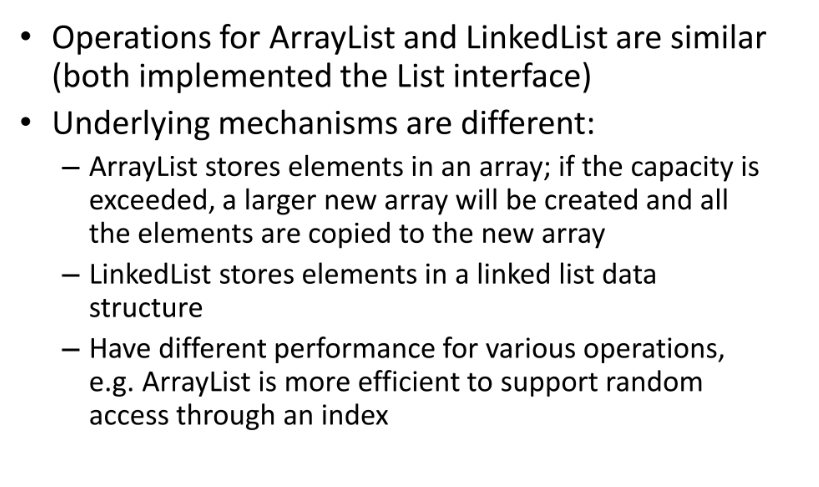
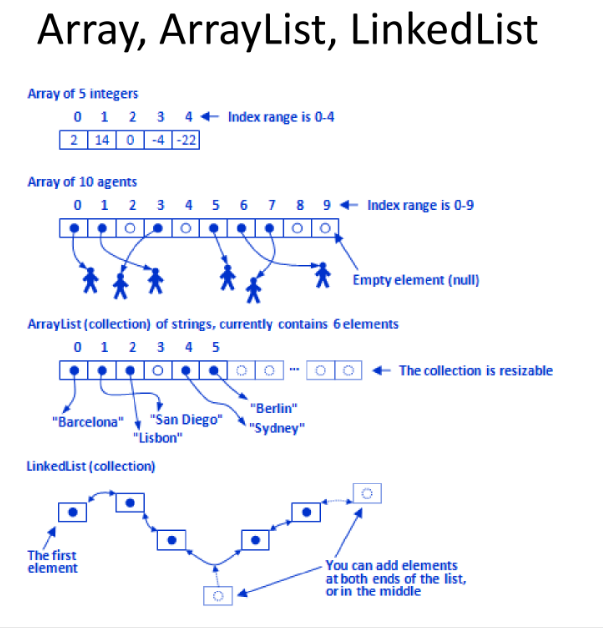
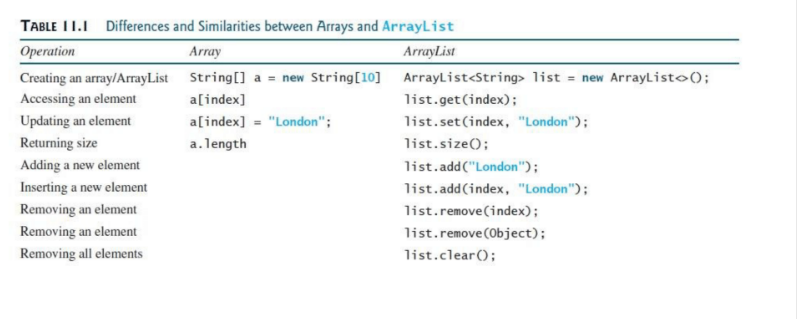


Two Different types in Java

1. Primitive type (Eg. Variables)
2. Reference type (Eg. Objects)



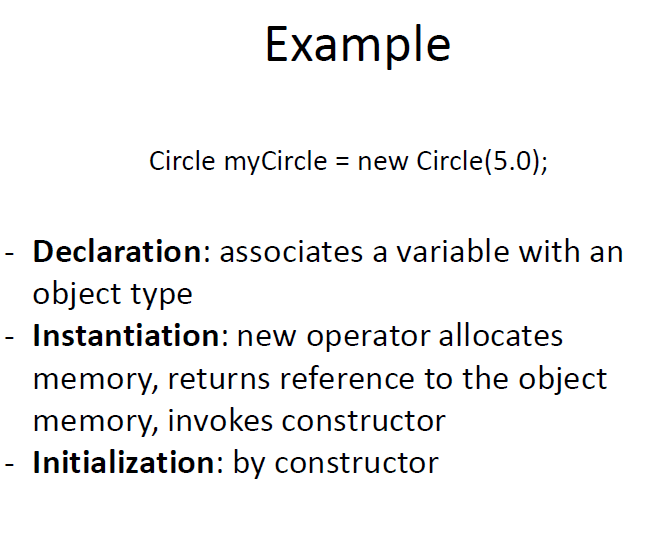
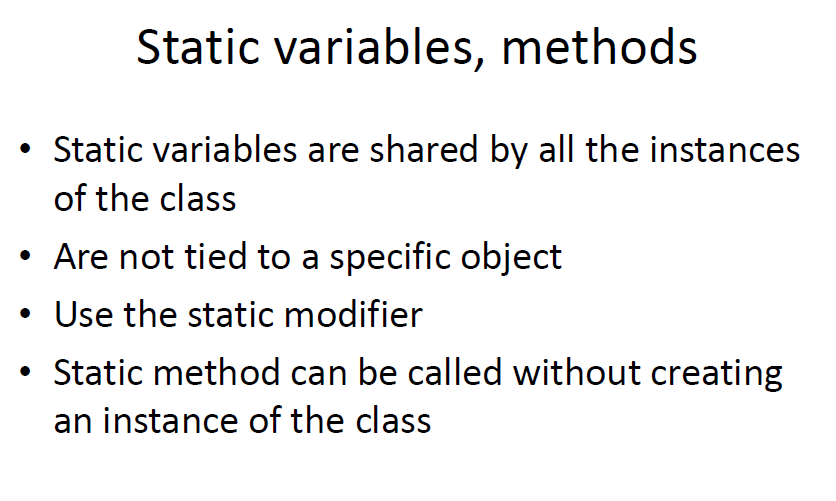
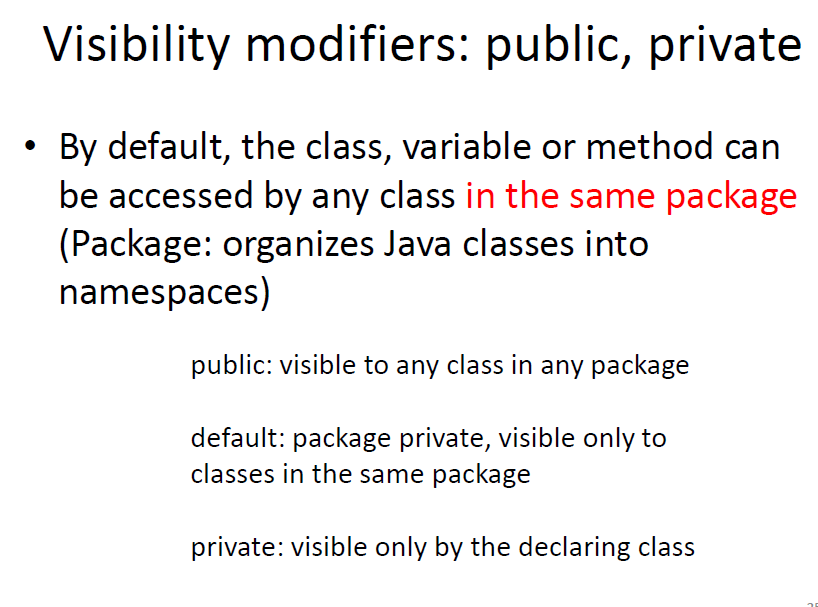
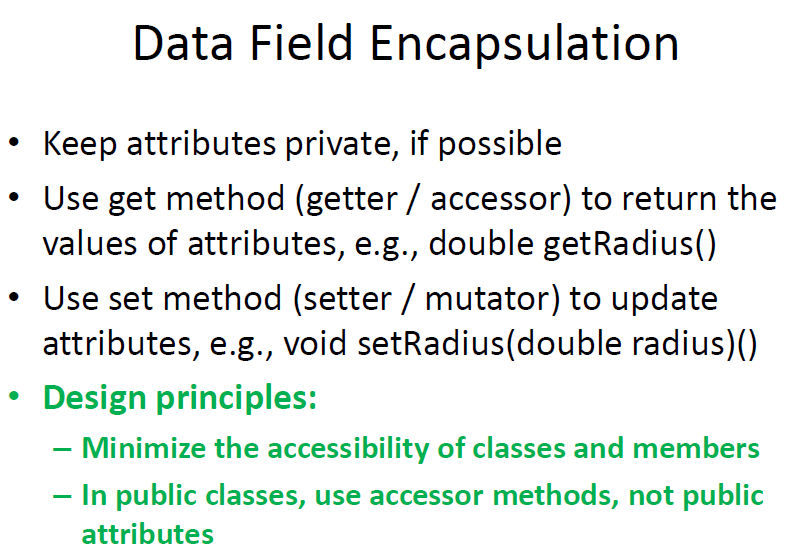
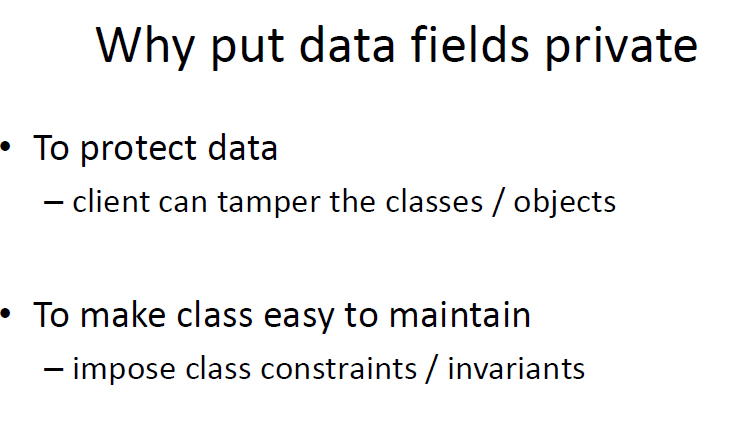
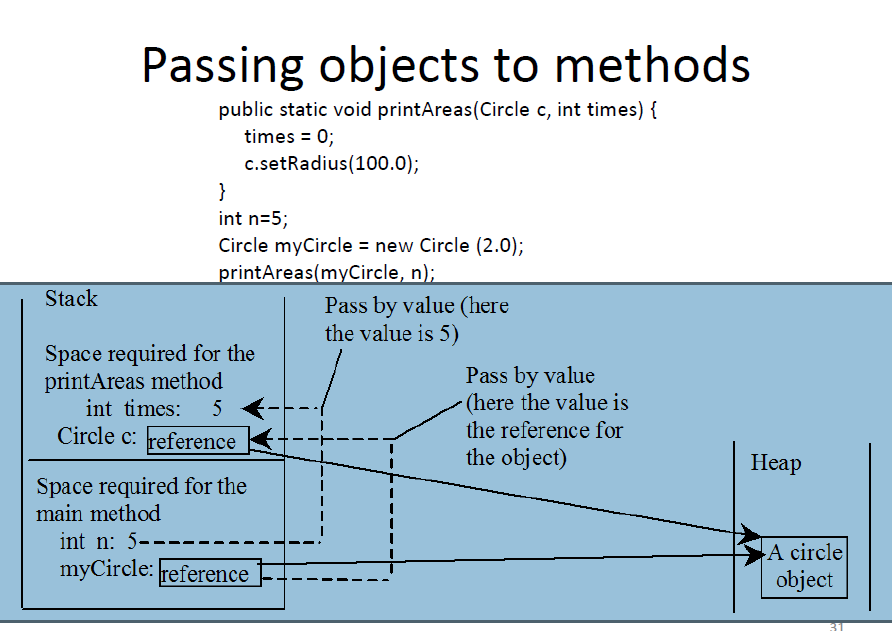
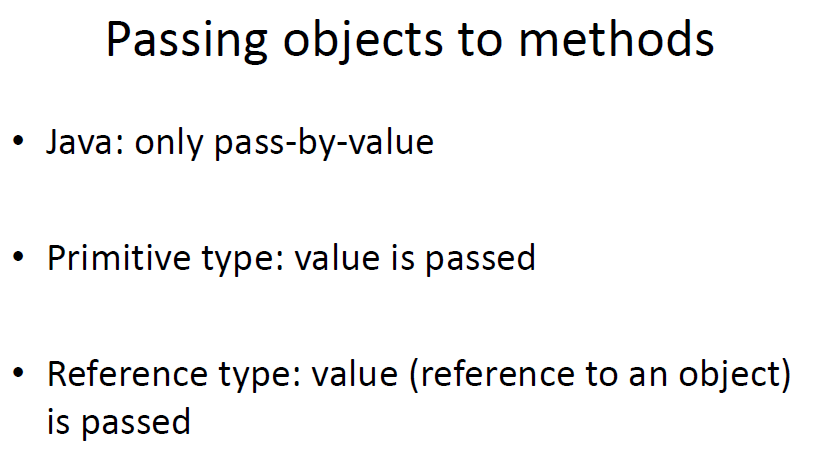


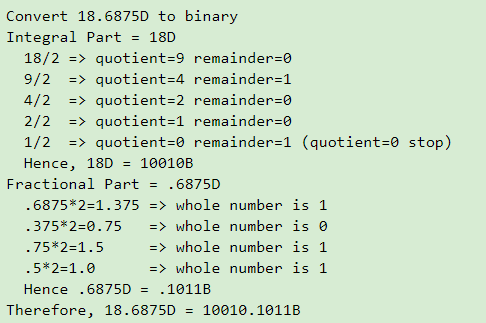
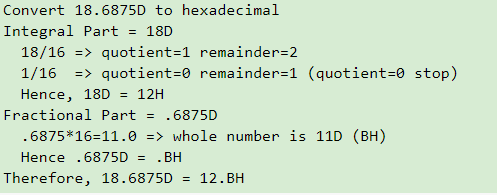
Array List and Linked List  


-

Random Access  
Array List Fast 0ms  
Linked List Slower 1978ms

Object Oriented Programing  
-**public class** Circle {  
 *//data* **double radius**;  
  
 *//constructor* **Circle(){**  
 **radius**=1;  
 }  
 Circle(**double** newRadius){ *//method overloading 2 methods...* **radius**=newRadius;  
 }  
  
 *//method* **double** getRadius(){  
 **return radius**;  
 }  
 **void** setRadius(**double** newRadius){  
 **radius**=newRadius;  
 }  
}

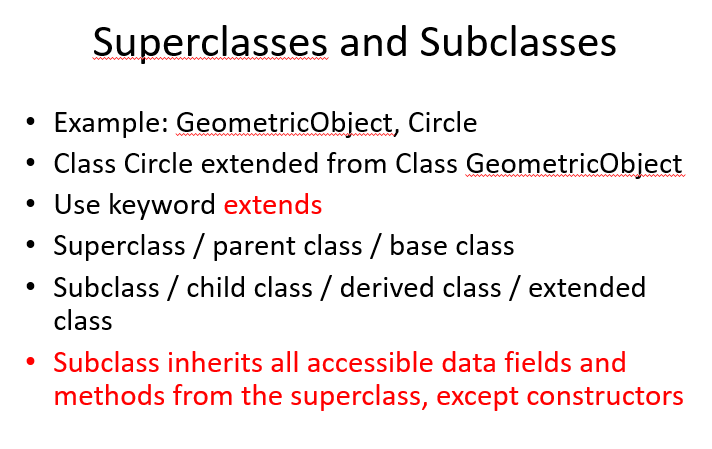
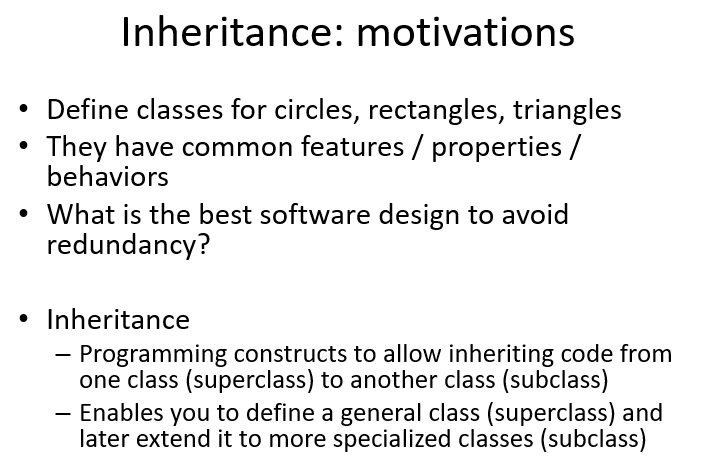
Constructor must have the same name as the Class.  
No return type. (Void or Int or etc)  
Circle(){…}  
Circle(int 123){…}   
  
  
  
**Instance variable and methods. Vs Static variable and static method.**(Static vs instance) P**S**VM  
It is unique to a unique object not class.  
account.interestRate vs Kenny.deposit(); For account is class, Kenny is the object.  
  
  
  
  
  
4  


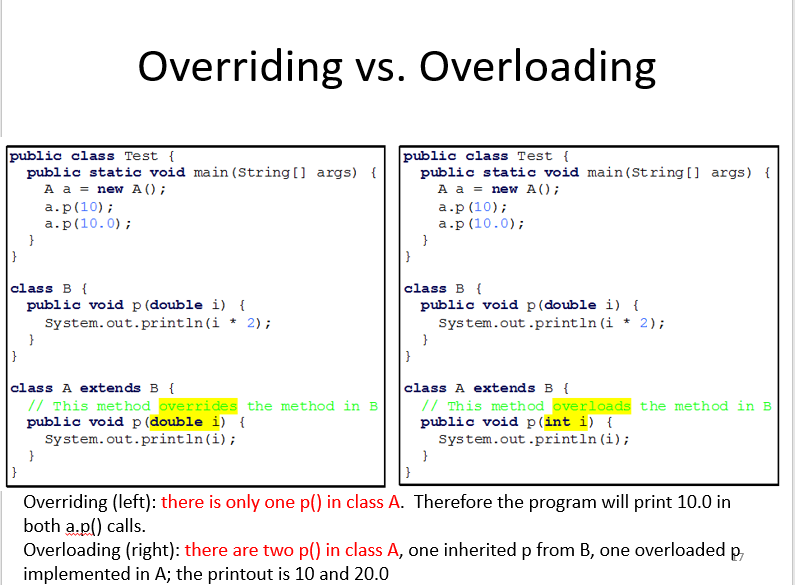
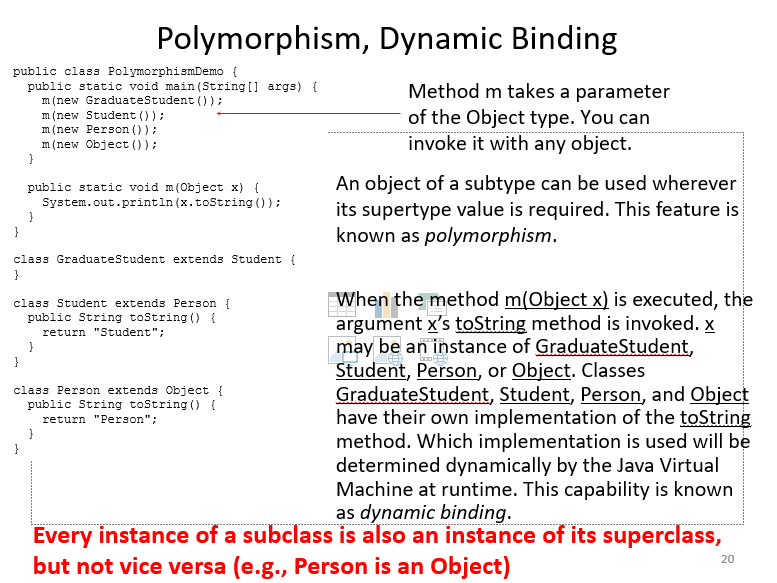
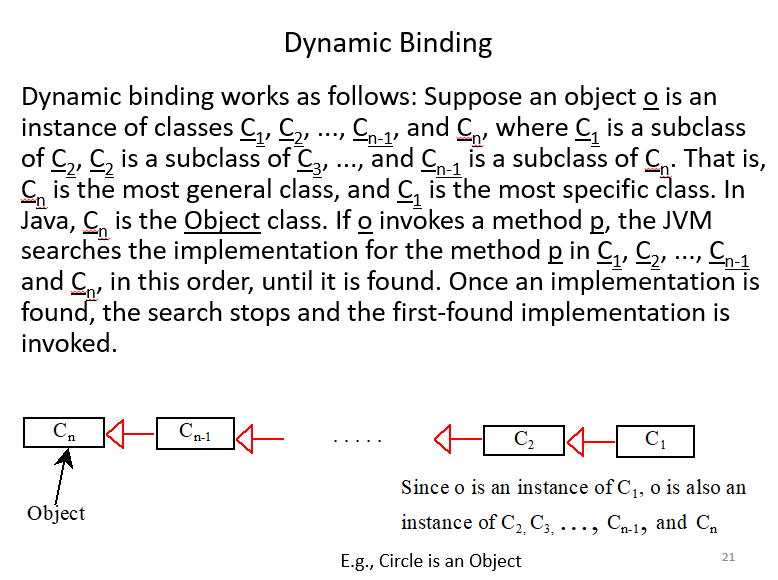
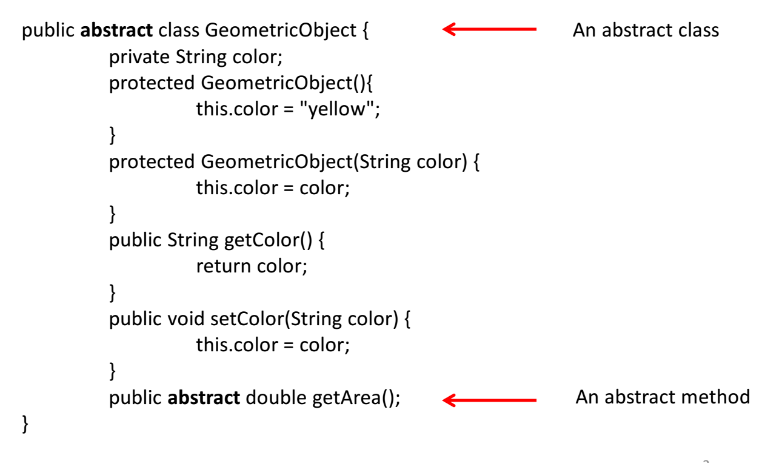
Week 1 Part B  
Change of base,  
Bits(base2) Hexadecimal(Base 16) Decimal(Base 10)  
0101000B 49AH 410D   
 0123456789ABCDEF  
 

<https://www.ntu.edu.sg/home/ehchua/programming/java/DataRepresentation.html>

Changing pi to a different base  
**3.416 can changed to base 10**3\*10^0+4\*10^-1+1\*10^-2….  
**3.416 can changed to base 10**3\*16^0 + 2\*16^-1….  
<https://www.youtube.com/watch?v=E1kOFHhNij4>  
  
piTerm()

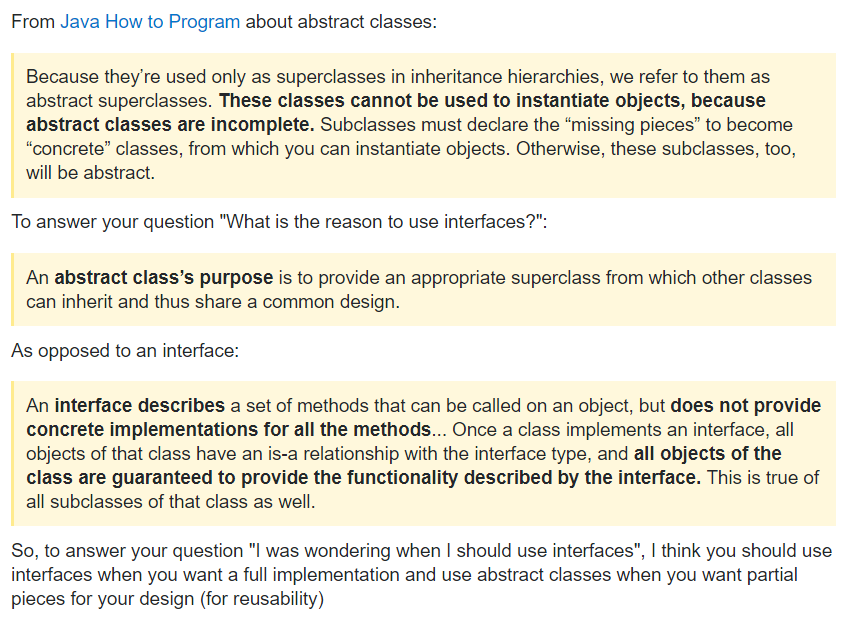
and piDigit()

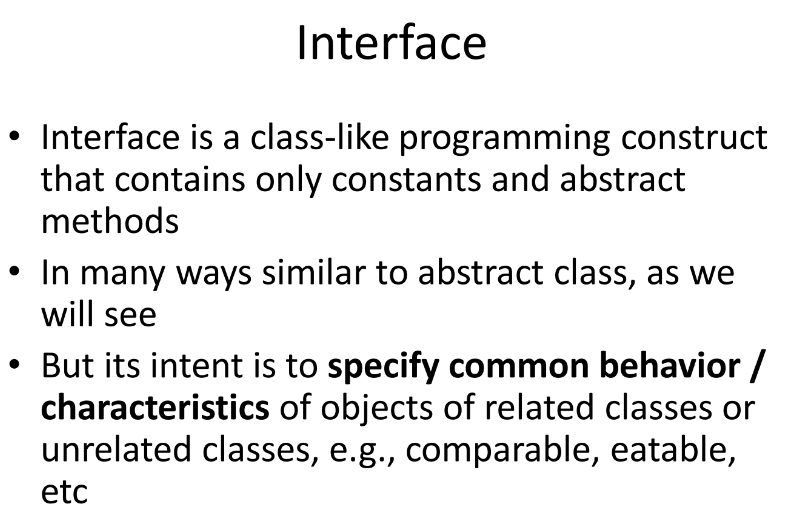
Week 2 Inheritance  
API (application interface)  
  
  
  
super(“color”), this(“color”) :   
super() calls super class constructor,   
this() calls the other overloaded constructor….when theres 2 constructor

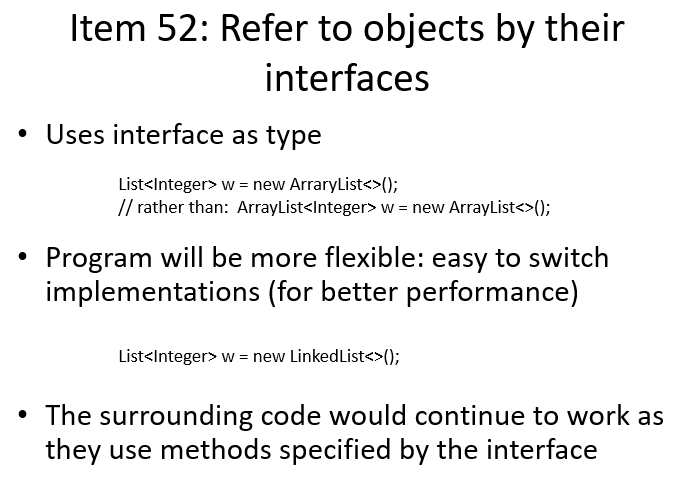
Overriding – overriding a method of a superclass on the subclass scipt  
Overloading – on the same script write 2 different method with same name so theres 2 methods.  
  
  
  
  
  
  
casting/ downcasting / The instance of Operator  
Actual type and declared type.  
  
GeoObject g= new Rect();  
GeoObject g1=new GeoObject();  
(Print override method of rect// can only use the declared types methods)  
  
Rect r= (Rect) g; //casting   
System.out.println(r.getHeight());  
  
**Week3**  
Concrete class vs abstract class (normal concrete can create an instance, Abstract cant create instance)  
Abstract Method- declare in super class but implementation is in subclasses.  


A subclass of abstract don’t have to implement all the abstract class methods.  
Interface needs to implement all.

An abstract class must have a constructor because once its subclass is implemented the abstract class is also constructed.

Week3 cohort activity for abstract class  
123456  
TTTFTT  
  
<https://stackoverflow.com/questions/16781329/when-do-i-have-to-use-interfaces-instead-of-abstract-classes>  
Abstract class & Interface are similar. (Abstract can provide method implementation but interface only act as a guide to tell subclasses what need to be implemented.

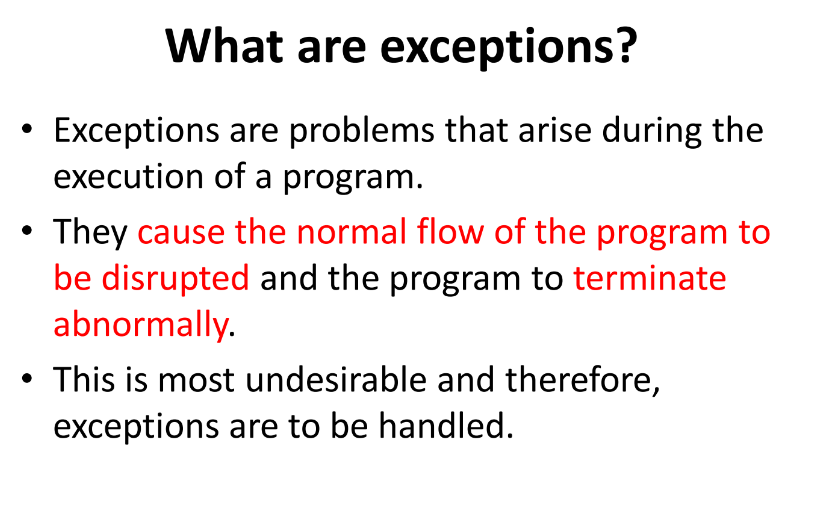


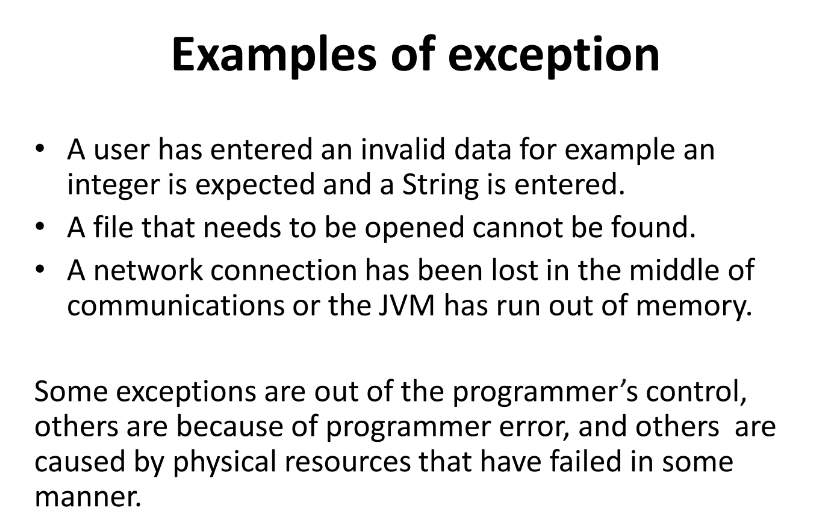
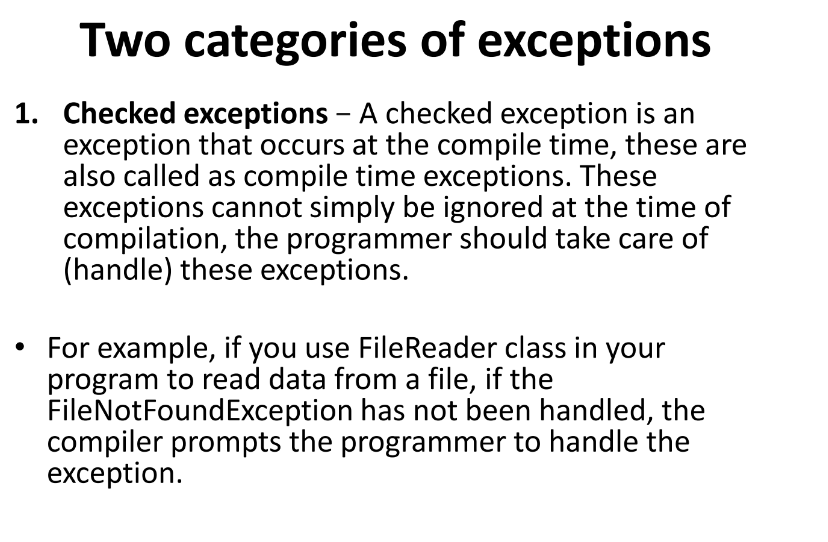
2d Array ( arrays in arrays)  
Arrays.deepToString(a)  
  
Some effective java code  
  


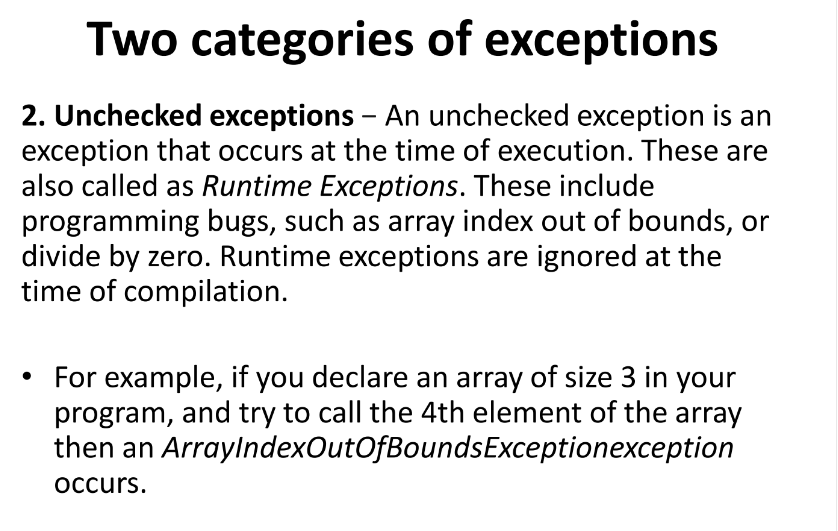
Regex expression  
public static void regexChecker   
Pattern checkRegex

Effective java   
Making it stable.

Immutable class( **final** makes it immutable means you cant change it anymore)  
private **final** Date startTime  
private final Date endTime

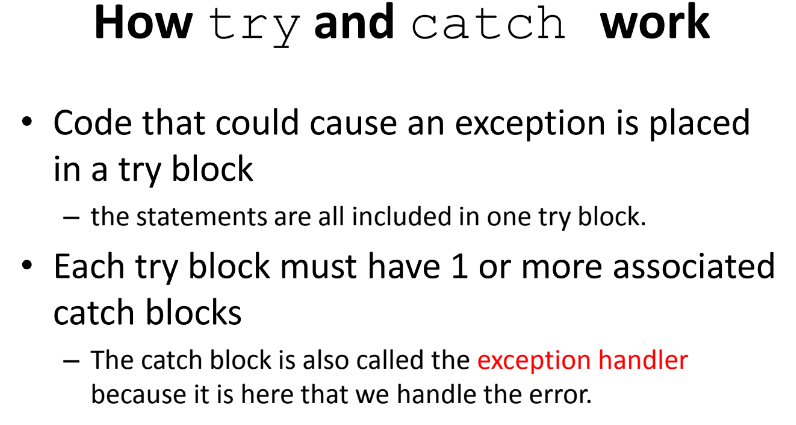


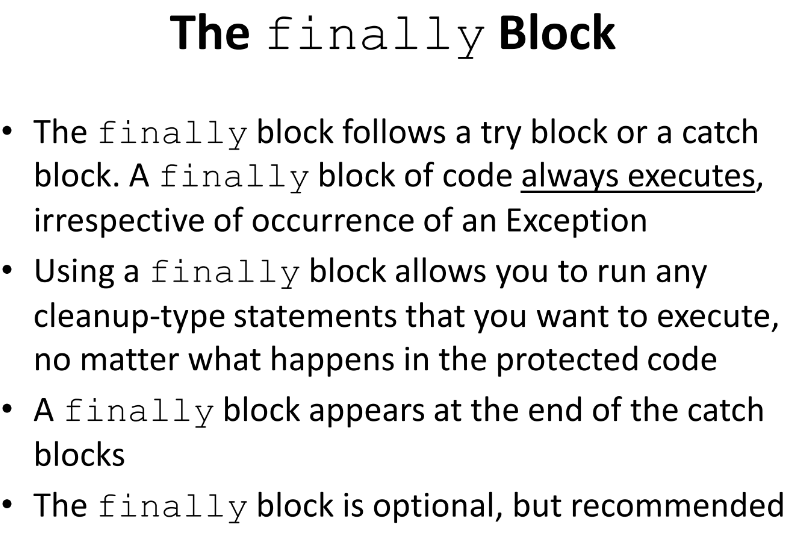
  
  
Exception handler is Catch block   
  


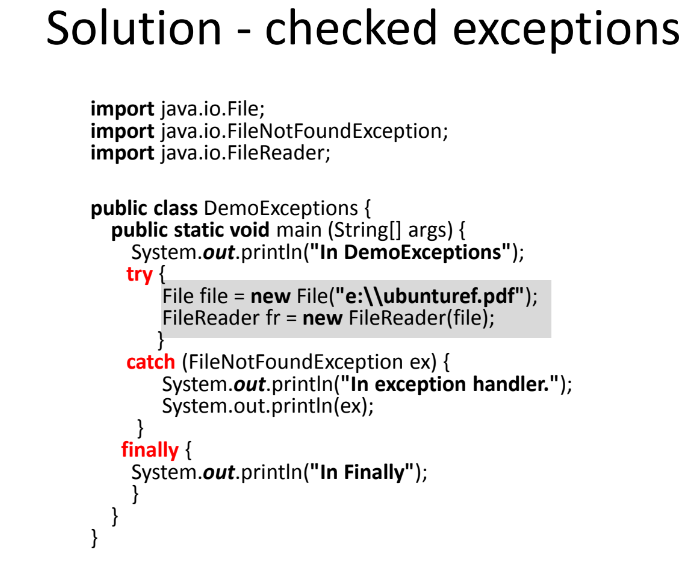
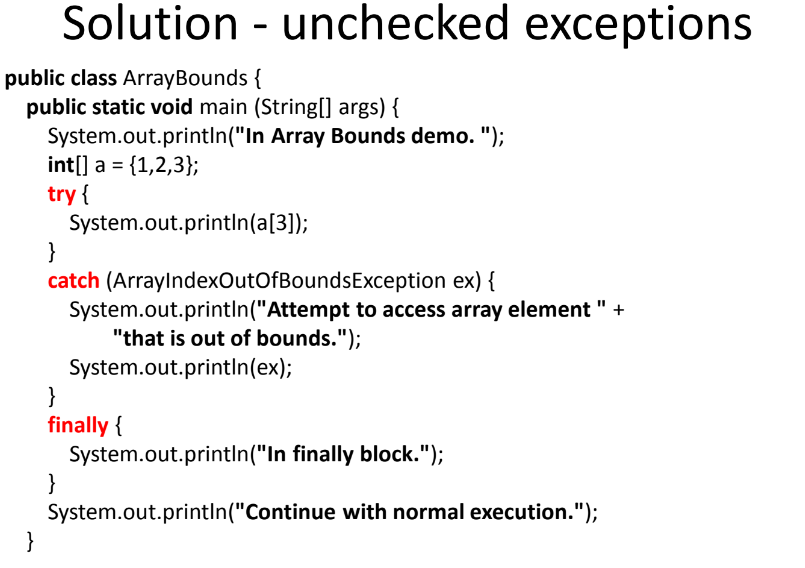
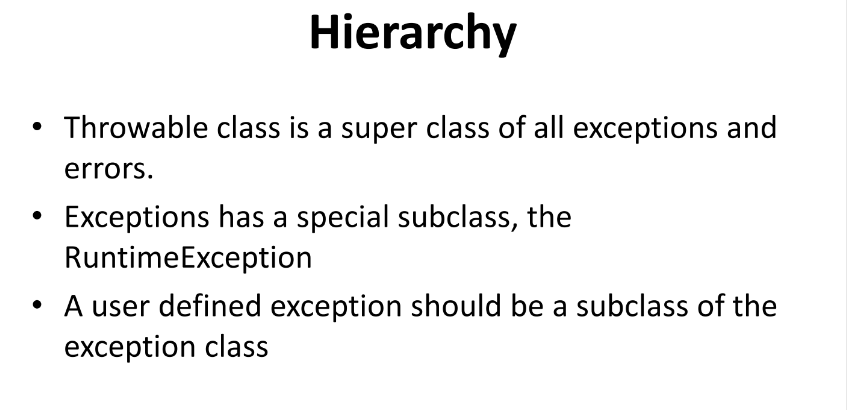
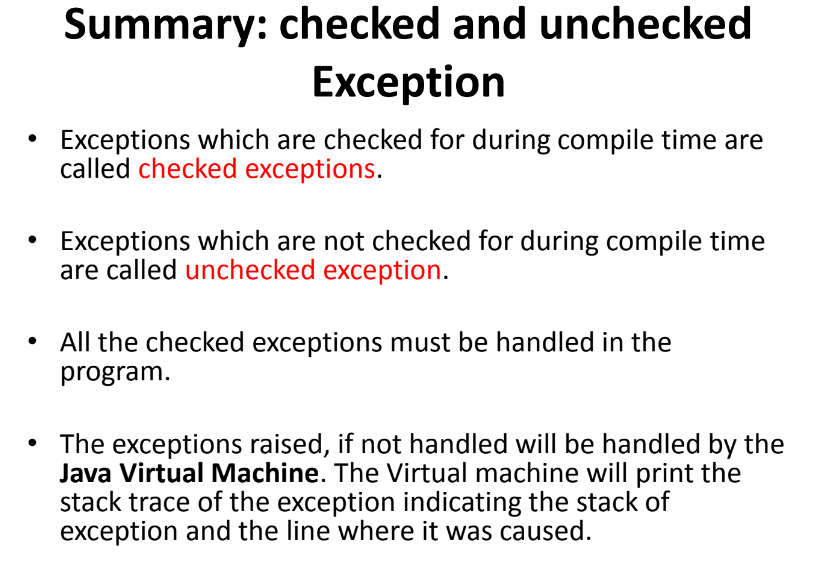


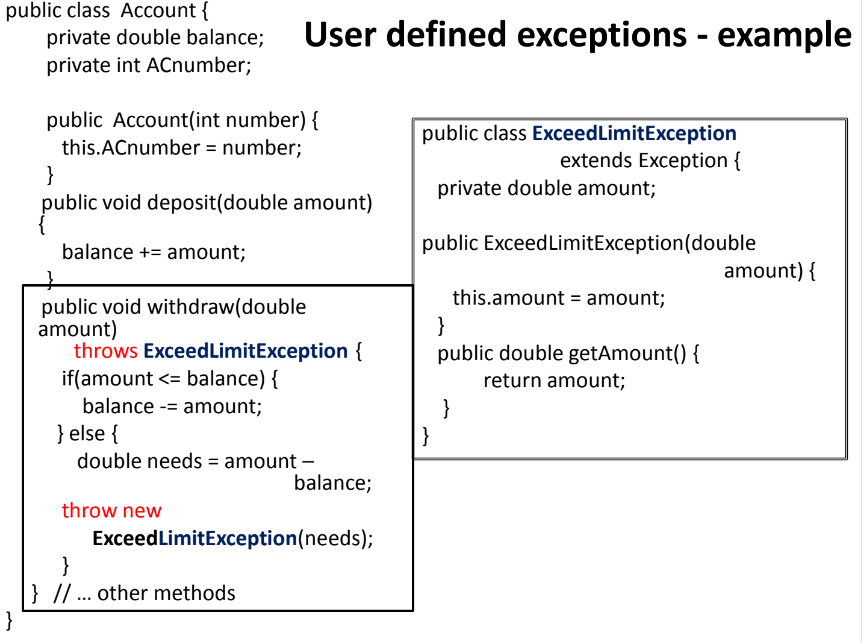
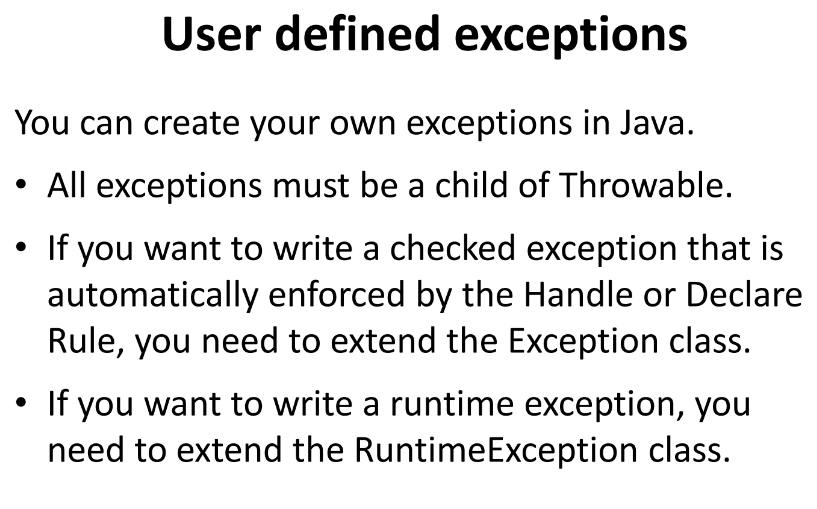
JVM has an exception that prints out ARRAY IS OUT OF BOUND.

If Try does not work does to Catch . (Try-> Catch) -> Finally Min: Try-> Catch (Blocks)







Exception is a parent class of index out of bound.

Anagram

Prefix and suffix  
Recursive call  
Rotate Left to create a rearranged suffix