/\*   
  
 File inputFile = new File( "input.txt" );   
 Scanner file = new Scanner( inputFile );   
   
 input and output streams  
 -java supports text and binary files  
 input classes  
 - inputstream -> abstract superclass represnting a stream of raw bytes  
 - fileinputstream -> input stream to read raw bytes of data from files  
 - objectInputStrea- class to read/recover from a file written using   
 ObjectOutputStream;  
 open/close streams  
 - void cose()  
 releases resources assocated with an open input stream thorws i/o excpetion  
 - system.in  
 - system.out  
 - system.err  
   
 broad overview of file i/o  
 use scanner class to establish io streams  
 use file class to establish connection wiht file  
 throws a checked exception filenotfoundexception  
 once a file is open for reading  
 - while loop as long as there is data to be read  
 -EITHER  
 -token by token using Scanner hasnext() or next()  
 - read one line as a string, prase with split()  
 -convert the tokens into individual scalers using wrapper class  
 conversion  
 writing out data  
 -use printwriter  
   
   
  
   
 /\*  
 o The difference between do-while and while is that do-while evaluates its expression   
 at the bottom of the loop instead of the top.   
 Therefore, the statements within the do block are always executed at least once,  
   
 }  
   
\*/

/\*  
 try(input == boolean c)  
 { exceptions  
 }  
 catch(blah) {  
 }  
   
 3 kinds of error   
 compilation  
 run time ---> exception which are run time error indicating something wrong  
 logic  
 2 components  
 excpetion classes: predifined classes for different errort ypes, EXCPETION is the super  
 exception handling blocks:   
 try: wraps code which may result in error  
 catch: contains exception handling statements  
 -catch ( SpecificExceptionType objSpecificExceptionType )   
 finally: statements executed whether or not an exception takes place (optional)  
 types of exceptions:  
 unchecked: do not have to be caught and handled by try/catch  
 if not handled code will compile and run but error condition will generate stack trace  
 and stop running  
 -runtimeexception or error  
 -execute wheter or not any exception was generated  
 -close what was opened, terminate what you need etc.  
 checked: must be caught and handled by try/catch blocks  
 -if not handled code will not compile  
 -io exceptions FILE EXCEPTIONS   
 exception class methods  
 -getMessage() returns message with cuase of exception  
 -toString() returns a String contaniing the exception class name and  
 message indication cause  
 -printStackTrace() prints the line number of code that cuased the exception  
 along with the sequence of method calls leading up to the exception   
 catching multiple exceptions:  
 -one catch block per exception type  
 - multiple catch blocks  
 -most specific to most general  
 -similar to switch where onnly the first matching excpetion gets executed  
 downsides  
 -not every problem needs it, slows down  
 -file i/o using scanner NEEDS TRY CATCH  
 -first handle bad data using simpler basic techniques  
 user defined exceptions  
 -part1: create a new exception class extending  
 -2:create a new class which uses the new exception class  
 -3: use the new class with some client code   
   
 public class NewExceptionClass extends ExistingExceptionClass {   
 public NewExceptionClass( String message ) {   
 super( message );   
 \*/  
C:\Users\varun\Documents\community college\csci 12\prgrm\FINAL