

```
PS C:\Users\saket\Git\CSWork\JAVA\Labs\HiddenMessagesLabP6BakshiSaket> java Main
meet at midnight
PS C:\Users\saket\Git\CSWork\JAVA\Labs\HiddenMessagesLabP6BakshiSaket>
```

The screenshot shows the AutoGradr web interface. On the left is a sidebar with navigation links: DASHBOARD > APCS-A-2018-P6 > HIDDEN MESSAGES, PROJECTS, Hidden Message, SAKET BAKSHI, Help / Docs, Password Change, and Logout. The main content area is titled 'Hidden Message' and includes a hint: 'Look at canvas Look at the input.. there will be no extra lines between cases...'. It features tabs for 'Extended Case 1', 'Hard Case', and 'Base Case'. A 'Console' section displays the input: '1 meet at midnight', '2 timsiovee netelt boi', and '3'. Below this is an 'Input Files' section with a link to 'input.txt' and a 'Help' button. A large drag-and-drop area is present with the text 'Drag and drop or browse'. At the bottom, a green notification bar states 'You passed 3 of 3 test cases', and a 'Passed' status is shown.

/* Saket Bakshi. Mr. Caces. AP Computer Science A. Due 27 January 2019.

This class takes messages and decodes them through a cover sheet, as done in the 90s action thriller, Con Air.

*/

```
public class HiddenMessagesLabP6BakshiSaket
```

```
{
```

```
    //instance variables
```

```
    private String[] originalMessage; //to hold the original message
```

```
    private String decodedMessage; //to hold the decoded message
```

```
    private int numberOfLinesInCover; //number of lines in the cover sheet
```

```
    private String[] coverLines; //content of each line in the cover sheet
```

```
    private int rowNum; //number of rows
```

```
    private int colNum; //number of columns
```

/** This class takes messages and decodes them through a cover sheet, as done in the 90s action thriller, Con Air

```
    @param message the original message
```

```
    @param numberOfLinesInCover the number of lines in the cover sheet
```

```
    @param coverContent an array of the cover, where each element contains a line of the array,
in order
```

```
    @param rowNum number of rows
```

```
    @param colNum number of columns
```

```
    */
```

```
    public HiddenMessagesLabP6BakshiSaket(String[] message, int numberOfLinesInCover,
String[] coverContent, int rowNum, int colNum)
```

```
{
```

```

this.originalMessage = message;
this.decodedMessage = "";

this.numberOfLinesInCover = numberOfLinesInCover;
this.coverLines = coverContent;
this.rowNum = rowNum;
this.colNum = colNum;
}

/** decodes the message
 */
public void decode()
{
    for(int i = 0; i < this.numberOfLinesInCover; i++) //goes through each line in cover
    {
        for(int k = 0; k < this.coverLines[i].length(); k++) //goes through each character in each line
        {
            String openOrNot = this.coverLines[i].substring(k, k+1);
            if(openOrNot.equals("O")) //checks if the character is "open"
            {
                this.decodedMessage = this.decodedMessage +
this.originalMessage[this.rowNum+i].charAt(k+this.colNum); //if open, the character at the same
spot in the message is added to the decoded message
            }
        }
    }
}

/** returns the decoded message
 @return the decoded message
 */
public String getDecodedMessage()
{
    return this.decodedMessage;
}
}

```

/* Saket Bakshi. Mr. Caces. AP Computer Science A. Due 27 January 2019.

This class takes inputs for the class, HiddenMessagesLabP6BakshiSaket, and prints out decoded messages.

*/

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Main

{

public static void main(String[] args) throws FileNotFoundException {

File file = new File("input.txt"); //brings input into console

Scanner scanned = new Scanner(file); //creates Scanner object out of input file

int numberOfCases = scanned.nextInt(); //takes number of cases

for(int m = 0; m < numberOfCases; m++) //makes loop for number of cases

{

int numLinesMessage = scanned.nextInt(); //takes number of lines in original message

scanned.nextLine(); //goes to next line that starts the original message

String[] originalMessage = new String[numLinesMessage]; //makes array to contain each line of original message

for(int i = 0; i < numLinesMessage; i++) //fills in array from previous line

{

originalMessage[i] = scanned.nextLine();

}

String startCoordinate = scanned.next(); //takes in start coordinate of cover

String rowNumberString = startCoordinate.substring(0, startCoordinate.indexOf(", "));

int rowNumber = Integer.parseInt(rowNumberString); //converts to integer for row

String columnNumberString = startCoordinate.substring(startCoordinate.indexOf(", ") + 1);

int columnNumber = Integer.parseInt(columnNumberString); //converts to integer for column

int numberOfLinesInCover = scanned.nextInt(); //takes number of lines in cover

scanned.nextLine(); //goes to next line which starts the cover

String[] coverContent = new String[numberOfLinesInCover]; //makes an array to hold each line of the cover

for(int i = 0; i < numberOfLinesInCover; i++) //fills in array from previous line

{

```
        coverContent[i] = scanned.nextLine();
    }

    HiddenMessagesLabP6BakshiSaket tester = new
HiddenMessagesLabP6BakshiSaket(originalMessage, numberOfLinesInCover, coverContent,
rowNumber, columnNumber); //creates HiddenMessages object
    tester.decode(); //decodes the message

    System.out.println(tester.getDecodedMessage()); //returns the message for each line in the
cover
    }
}
}
```