1. Homework 1

Posted: August/10/2014
Due: August/31/2014 24.00

Homework 1.1, 1.2, and 1.3 are due August/31/2014 24.00.

Most of the lab sections taught in the CS Department use the try command to allow students to electronically submit their work.

**IMPORTANT:** EVERY STUDENT has to register.

**IMPORTANT:** EVERY STUDENT has to submit a solution for hw 1.1.

All submission and registrations must be done from glados.cs.rit.edu. To register with *try*, open a terminal window and from a Unix prompt and type the following command:

```
ssh glados.cs.rit.edu
try hpb-grd register /dev/null
```

The program should run like this:

```
try hpb-grd register /dev/null
Copying files...done
Section Time(s) & Room(s)  Instructor
-------------------------------
01  MW 10:00-11:50  23-1320  Bischof
```

We are recording the following information for you:

Permanent account: hpb
Temporary account:
Lecture section:  70
Lab section:  1
Name:

If any of this information is incorrect, see your instructor.
You are registered.

You can not submit any homework before you have yourself registered!

1.1. Programming Standard

Please make sure that your code is compatible with the
We will take points of, if this is not the case.

1.2. Submission

You can submit a solution as often as you like, but only the last submission will be stored.
Copy the solutions to glados.cs.rit.edu.
Log on to glados via *putty* or *ssh*. 
# shown for ssh
% ssh glados.cs.rit.edu
% cd where_your_solution_is_stored
% try hpb-grd lab1-3 Board.java
Copying files...done

ON-TIME submission of lab1-3
Not Compiling your program...

Files being saved:
Board.java

lab1-3 has been submitted.
%

1.3. Teamwork

Homework 1.1 has to be done by every student. All other homeworks have to be submitted as a team of 2.

Homework 1.1 (10 Points)

Look at the Hello program:

```java
/**
 * Classical: Hello, World
 * *
 * @version $Id: Hello.java,v 1.3 2001/06/06 23:05:46 hpb Exp hpb $
 * @author hpb
 * *
 * Revisions:
 * *
 * Revision 1.41 2013/06/06 16:19:12 hpb
 * Revision 1.42 2014/08/06 10:31:21 hpb
 * Initial revision
 * *
 */

class Hello {
    public static void main (String args []) { // main program
        System.out.println("Hello World!");
    }
}
```

Source Code: Src/3/Hello.java

and modify it in a way, that it prints out the operating system name. It is not allowed to hard code the os name. This is an output example if I execute the program:
% java OsName.java
OS: OS X

Hints: Strings can be concatenated with +. Look at
and

The following example shows how System.getProperties can be used:
This program list properties:

```java
import java.util.Enumeration;
import java.util.Properties;

class TestProperty {
    public static void main (String args []) { // main program
        Enumeration allProp = System.getProperties().propertyNames();
        while (allProp.hasMoreElements()) {
            String s = (String)allProp.nextElement();
            System.out.println(s + "	" + System.getProperty(s) + "!");
        }
        System.out.println(System.getProperty("user.dir"));
    }
}
```

Source Code: Src/21/TestProperty.java

You have to name the file: OsName.java
Submit your solution via:

% ssh glados.cs.rit.edu
# password
# go to the directory where your solution is ...
try hpb-grd lab1-1 OsName.java

Solution:
class OsName {
    public static void main (String args []) { // main program
        System.out.println("OS: "+System.getProperty("os.name") +".");
    }
}

Source Code: Src/21/OsName.java

** Homework 1.2 (10 Points)**

In how many different combinations can n objects be arranged? Assume the objects are named: a, b, and c. All possible combinations are: {}, {a}, {ab}, {ac}, {b}, {bc}, {c}, {abc}.

Write a program which creates all possible combinations. n should be hard coded in your program.

You have to name the file: Combination.java

Connect to glados.cs.rit.edu and go in the directory containing the solutions for hw1.

Submit your solution via:

```
try hpb-grd lab1-2 Combination.java
```

Example:

```
% java Combination.java
{ {}, {a}, {b}, {ba}, {c}, {ca}, {cb}, {cba}, {d}, {da}, {db}, {dba}, {dc}, {dca}, {dcb},
```

Solution:

```java
/**
 * Combination.java: See proof
 * Sn = { s1, s2, s3, ... sn }
 * Lemma: | AllSubeSetsOfSn | = 2^n
 */

* Proof by induction
* n = 0: | AllSubeSetsOfSn0 | = | { {} } | = 1 = 2^0
* n -> n + 1: Sn+1 = { s1, s2, s3, ... sn, sn+1 }
* SAllSubeSetsOfSn = { s1, ... s2^n } 
* AllSubeSetsOfSn+1 = { s1, ... s2^n } U { s1, ... s2^n }
* |AllSubeSetsOfSn+1 | = | { s1, ... s2^n } | + | { s1, ... s2^n } |
* |AllSubeSetsOfSn+1 | = 2^n + 2^n
* |AllSubeSetsOfSn+1 | = 2 * 2^n
* |AllSubeSetsOfSn+1 | = 2^(n+1)

* Solution Idea:
* if bit at position i is 1 print argument[i]
* else, print nothing
```
* For example:
  * 0 = {} 0
  * 1 = {} a 0 1
  * 2 = {} a b ab 0 01 10 11
  * 3 = {} a b c ab ac bc abc 0 01 10 11
  * ...

```java
public class Combination {

    static String[] args;
    static int soManyArgs;
    static int setSize;
    static int soManyBits;
    static int soManyBitsInAnByte = 8;

    private static void printSetForIndex (int value) {
        System.out.print("{ ");
        for (int index = soManyBits; index >= 0; index --) {
            if ((1 << index) & value) // bitmask
                System.out.print(args[index]);
        }
        System.out.print("}");
    }

    private static void doCombination() {
        soManyArgs = args.length;
        setSize = (int)Math.pow(2, soManyArgs); // empty set is represented as 0
        soManyBits = args.length; // only the last bits count
        System.out.print("{ ");
        for (int index = 0; index < setSize; index ++ ) {
            printSetForIndex(index); // using an int type limits the number of subsets
            System.out.print(index == setSize - 1? "}\n" : ", "); // newLine
        }
    }

    public static void main( String[] arguments ) {
        args = arguments;
        doCombination();
    }
}
```

Source Code: Src/21/Combination.java

Hints: Strings can be concatenated with +.
Homework 1.3 (10 Points)

Write a program that calculates the maximum number of kings that can be placed on the chessboard below, without threatening each other. You have to use a recursive algorithm. You can not test your program by running it to completion, because this might take too long.

Key:
- red/orange: board; you can place a king here
- white: not part of the board, you can not place a king here.
- green: there is already one king on this place; the king can not be moved. Example:

% java Board
I am not telling, but the output is a number

Submit your solution via:

try hpb-grd lab1-3 Board.java

Solution:

```java
public class Board {

    final int MAX_ROWS = 10;
    final int MAX_COLUMNS = 10;
    final Color KING_COLOR = Color.green;
    final Color NOT_LEGAL = Color.white;
    JButton theBoard[][] = new JButton[1+MAX_ROWS][1+MAX_COLUMNS];
    int soManyKings = 0;
    int maxNofKings = 0;

    public void initBoard(GridLayout gridbag, JPanel aPanel) {
        for (int row = 0; row < MAX_ROWS; row++) {
            for (int column = 0; column < MAX_COLUMNS; column++) {
                theBoard[row][column] = new JButton();
                Color color = (row + column) % 2 == 0 ? Color.orange : Color.red;
                theBoard[row][column].setBackground(color);
            }
        }
    }
```
aPanel.add(theBoard[row][column]);

for (int index = 1; index < MAX_ROWS-1; index++) {
    theBoard[1][index].setBackground(NOT_LEGAL);
    theBoard[MAX_ROWS-2][index].setBackground(NOT_LEGAL);
}

theBoard[0][4].setBackground(NOT_LEGAL);
theBoard[0][5].setBackground(NOT_LEGAL);
theBoard[MAX_ROWS-1][4].setBackground(NOT_LEGAL);
theBoard[MAX_ROWS-1][5].setBackground(NOT_LEGAL);
theBoard[4][0].setBackground(NOT_LEGAL);
theBoard[5][0].setBackground(NOT_LEGAL);
theBoard[4][MAX_COLUMNS-1].setBackground(NOT_LEGAL);
theBoard[MAX_ROWS-1][MAX_COLUMNS-1].setBackground(NOT_LEGAL);

theBoard[4][4].setBackground(NOT_LEGAL);
theBoard[4][5].setBackground(NOT_LEGAL);
theBoard[5][5].setBackground(NOT_LEGAL);
theBoard[5][4].setBackground(NOT_LEGAL);

}

public void setPieceTo(int row, int column) {
    Color aColor = theBoard[row][column].getBackground();
    if ((aColor == Color.orange) || (aColor == Color.red) || (aColor == KING_COLOR))
        theBoard[row][column].setBackground(KING_COLOR);
}

public void removePieceFrom(int row, int column) {
    Color aColor = theBoard[row][column].getBackground();
    Color newColor = ((row + column) % 2 == 0 ? Color.orange : Color.red);
    if (aColor == KING_COLOR)
        theBoard[row][column].setBackground(newColor);
}

public boolean isThereAking(int row, int column) {
    if (row < 0)
        return true;
    if (column < 0)
        return true;
    if (row >= MAX_ROWS)
        return true;
    if (column >= MAX_COLUMNS)
        return true;

    Color aColor = theBoard[row][column].getBackground();
    return (aColor != KING_COLOR);
}
public Component createComponents() {
    JPanel aPanel = new JPanel();
    GridLayout grid = new GridLayout(MAX_ROWS, MAX_COLUMNS);
    grid.setHgap(2);
    grid.setVgap(2);
    aPanel.setLayout(grid);
    initBoard(grid, aPanel);
    setPieceTo(3, 3);
    setPieceTo(3, 6);
    setPieceTo(6, 3);
    setPieceTo(6, 6);
    
    /*
     * setPieceTo(0, 0);
     * setPieceTo(0, 1);
     * setPieceTo(1, 2);
     * setPieceTo(9, 9);
     * removePieceFrom(0, 0);
     * removePieceFrom(0, 1);
     */
    return aPanel;
}

public void waitForAWhile() {
    try {
        Thread.sleep(100);
    } catch (InterruptedException ie) {
    }
}

/*
 * ----> column
 * |
 * |
 * v
 * row
 * 
 * --+----+----+----+--------
 * |1 |8 |7 |
 * --+----+----+----+--------
 * |2 |6 |
 * --+----+----+----+--------
 * |3 |4 |5 |
 * --+----+----+----+--------
 */

public boolean isSave(int thisRow, int thisColumn) {
    boolean isSave = true;
    isSave = isThereAking(thisRow - 0, thisColumn - 0 )
    isThereAking(thisRow - 1, thisColumn - 1 )
    isThereAking(thisRow - 0, thisColumn - 1 )
    isThereAking(thisRow + 1, thisColumn - 1 )
    isThereAking(thisRow + 1, thisColumn + 0 )
    isThereAking(thisRow + 1, thisColumn + 1 )
}
```java
43             isThereAKing(thisRow - 0, thisColumn + 1)
44             isThereAKing(thisRow - 1, thisColumn + 1)
45             isThereAKing(thisRow - 1, thisColumn - 0);
46
47             return isSave;
48         }
49     public void testKings() {
50         setPieceTo(MAX_ROWS -1, MAX_COLUMNS -1);
51         setPieceTo(MAX_ROWS -1, MAX_COLUMNS -2);
52         int r = MAX_ROWS -1;
53         int c = MAX_ROWS -3;
54         System.out.println("isSave " + r + "/" + c + ": " + isSave(r, c));
55     }
56
57     public void placeKings(int startC, int startR) {
58         for (int row = startR; row < MAX_ROWS; row ++ ) {
59             for (int column = startC; column < MAX_COLUMNS; column ++ ) {
60                 waitForAWhile();
61                 Color aColor = theBoard[row][column].getBackground();
62                 if ( aColor == NOT_LEGAL )
63                     continue;
64                 if ( isSave(row, column) ) {
65                     // System.out.println(" add king at: " + row + "/" +
66                     setPieceTo(row, column);
67                     soManyKings ++;
68                     if ( soManyKings > maxNofKings )
69                         maxNofKings = soManyKings;
70                     if ( column == MAX_COLUMNS - 1 )
71                         placeKings(row + 1, 0);
72                     else
73                         placeKings(row, column + 1);
74                     soManyKings --;
75                 }
76             }
77     }
78
79     }
80     public void printMaxKings() {
81         System.out.println("maxNofKings = " + maxNofKings);
82     }
83
84     public static void main(String[] args) {
85         String lookAndFeel = UIManager.getCrossPlatformLookAndFeelClassName();
86         try {
87             UIManager.setLookAndFeel( lookAndFeel);
88         } catch (Exception e) { }
89         JFrame frame = new JFrame("Board");
90         frame.setPreferredSize(new Dimension(500,500));
91         Board app = new Board();
92         Component contents = app.createComponent();
```
frame.getContentPane().add(contents);

frame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
});

frame.pack();

frame.setVisible(true);

app.testKings();
app.placeKings(0, 0);

//app.printMaxKings();

Source Code: Src/21/Board.java

1.4. Extra Homework

Write a program which calculates \( \ln(2) \). You can not use any exciting method which can calculate \( \ln(2) \).