CS1110 3 Nov 2009 Testing/Debugging. Also, about A6

Read chapter 14, pp. 385–401

Prof. Lee is out of town, giving an invited lecture at CIS, Penn:
http://www.cis.upenn.edu/departmental/events/Lillian.lee.shtml

Title: A tempest: Or, on the flood of interest in sentiment analysis, opinion mining, and the computational treatment of subjective language

Royal Flush is better than Full House

Two-dimensional arrays

	0	1	2	3	b.length
b	5	4	7	3	

one-dimensional array

d		0	1	2	3
	0	5	4	7	3
	1	4	8	9	7
	2	5	1	2	3

rectangular array: 5 rows and 4 columns

Type of d is **int**[][] ("**int** array array", "an array of **int** arrays")

To declare variable d:
int d[][].

To create a new array and assign it to d:
d= **new int**[3][4];

To reference element at row r column c:
d[r][c]

number of rows

number of cols

A 2-dimensional array b

P00 P01 P02 P03
P10 P11 P12 P13
P20 P21 P22 P23

Same array in row-major order (rmo) c
P00 P01 P02 P03 P10 P11 P12 P13 P20 P21 P22 P23

You can see that b[i][j]
is same as c[i * (no of columns) + j]

Hiding character k (integer representation is 107) in a pixel

R: 254
G: 119
B: 034

→

R: 251
G: 110
B: 037

Manipulating jpg files

Subclass of JFrame. Has buttons, panels, etc. Has methods that are called when a button is clicked. Contains ImagePanel variables for original and current images

ImageGUI

ImagePanel

ImageProcessor

ImageArray

JPanel: contains an image. Its method paint draws the image.

Contains ImageArray variables for the original and current image. Has the methods for manipulating an image

Contains an array of pixels of an image, in row-major order. Has methods for getting, setting a pixel of the image,

Testing: Read chapter 14.

Bug: Error in a program.

Testing: Process of analyzing, running program, looking for bugs.

Test case: A set of input values, together with the expected output.

Debugging: Process of finding a bug and removing it.

Exceptions: When an error occurs, like divide by 0, or s.charAt[i] when i = – 1, Java *throws an exception*. A lot — generally too much — information is provided.

Exceptions: When an error occurs, like divide by 0, or s.charAt[i] when i = – 1, Java *throws an exception*.

```
06 /** = String s truncated …. */
07  public static String truncate5(String s) {
08      int b= 10 / 0;
09      if (s.length() <= 5)
10          return s;
11      return s.substring(0,5);
12  }
```

Turn on line numbering in DrJava. Preferences / Display Options

important part

```
ArithmeticException: / by zero
 at A4Methods.truncate5(A4Methods.java:8)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(….java:39)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(….java:25)
 at java.lang.reflect.Method.invoke(Method.java:585)
```

call stack

Debugging a program

When an error occurs, you have to play detective and find it. That process is called debugging. The place where the bug is may be far removed from the place where an error is revealed.

Strategy 0: Find a simplest possible test case that exhibits the error.

Strategy 1: put print statements, suitably annotated, at judiciously chosen places in the program.

Strategy 2: Use Java assert-statements at good places:
assert <boolean expression> ;

Strategy 3: Use the debugging feature of your IDE (Interactive Development Environment —yours is DrJava.

7

Assert statement

Use it to program "defensively", and leave it in the program

Example: Use it to check preconditions:

```
/** = "This Virus is the predecessor of v".
      Precondition: v is not null */
public boolean isPredecessorOf(Virus v) {
  assert  v != null;
  …
}
```

8

Debugging a program

When an error occurs, play detective and find it. Called debugging. The place where the bug is may be far removed from the place where an error is revealed.

```
public static HSV RGB2HSV(Color rgb) {
  …
  /**Set MAX, MIN to max and min of R, G, B */
  double MAX= 0; double MIN=  0;
  if (R>G && R>B)     {MAX= R; }
  if (G>B && G>R)     {MAX= G;}
  if (B>R && B>G)     {MAX= B;}
  if (R<G && R<B)     {MIN= R; }
  if (G<B && G<R)     {MIN= G; }
  if (B<R && B<G)     {MIN= B;}
  System.out.println("R " + R + ", G " + G +
                     ", B " + B + ", MAX " + MAX);
```

If you just output the numbers without naming them, you will have trouble.

9

Debugging a program

When an error occurs, play detective and find it. Called debugging. The place where the bug is may be far removed from the place where an error is revealed.

```
public static HSV RGB2HSV(Color rgb) {
  …
  /**Set MAX, MIN to max and min of R, G, B */
  double MAX= 0; double MIN=  0;
  if (R>G && R>B)     {MAX= R; }
  if (G>B && G>R)     {MAX= G;}
  if (B>R && B>G)     {MAX= B;}
  if (R<G && R<B)     {MIN= R; }
  if (G<B && G<R)     {MIN= G; }
  if (B<R && B<G)     {MIN= B;}
  assert R <= MAX  &&  G <= MAX  &&  B <=  MAX;
  assert MIN <= R  &&  MIN <= G   &&  MIN <=  B;
```

These assert statements don't check completely that MAX is the max and MIN the min.

10

```
public static HSV RGB2HSV(Color rgb) {
  …
  if (R>G && R>B)     {MAX= R; }
  if (G>B && G>R)     {MAX= G;}
  if (B>R && B>G)     {MAX= B;}
  if (R<G && R<B)     {MIN= R; }
  if (G<B && G<R)     {MIN= G; }
  if (B<R && B<G)     {MIN= B;}
  System.out.println("R " + R + ", G " + G +
                     ", B " + B + ", MAX " + MAX);
```

call and output

> A4Methods.RGB2HSV(new java.awt.Color(255,255,128))
R 1.0, G 1.0, B 0.502, MAX 0.0

Look! MAX is 0 and not 1!

if conditions should be >= , not >

11

```
…
if (Hi ==0){
     R=(int)(v  * 255.0);
     G=(int)(t  * 255.0);
     B=(int)(p  * 255.0);
}
if (Hi==1){
     R=(int)(q  * 255.0);
     G=(int)(v  * 255.0);
     B=(int)(p  * 255.0);
}
…
System.out.println("In HSVtoRGB. R is " + R);
int r= (int)Math.round(R);
System.out.println("In HSVtoRGB. r is " + r);
```

Error in HSVtoRGB. Not rounding properly

Insert println statements.

12