Embedded Content Image

Royal Flush is better than Full House

Two-dimensional arrays

- 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

- ImageGUI: 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
- ImagePanel: contains an image. Its method paint draws the image.
- ImageProcessor: Contains ImageArray variables for the original and current image. Has the methods for manipulating an image
- ImageArray: Contains an array of pixels of an image, in row-major order. Has methods for getting, setting a pixel of the image.

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

```java
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 }
```

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.

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.

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.

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.

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.
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:

\[
\text{assert } \text{boolean expression};
\]

Example: Use it to check preconditions:

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

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.

```java
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);
    ...
    Error in HSVtoRGB. Not rounding properly
    System.out.println("In HSVtoRGB. R is " + R);
    System.out.println("In HSVtoRGB. r is " + r);
}
```

```java
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);
    ...
    Error in HSVtoRGB. Not rounding properly
    System.out.println("In HSVtoRGB. R is " + R);
    System.out.println("In HSVtoRGB. r is " + r);
}
```

```java
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);
    ...
    Error in HSVtoRGB. Not rounding properly
    System.out.println("In HSVtoRGB. R is " + R);
    System.out.println("In HSVtoRGB. r is " + r);
}
```