CS1110 11 November Testing/Debugging. And Applications

Read chapter 14, pp. 385–401

Prelim 2
Max 100
Median 82
Mean 78.3
Min 20

TAs can fix obvious mistakes.
Real regrade request? Write note explaining the issues, attach to prelim, give to Gries or Lee before Thanksgiving.

```
100      1 *
90-99  53 ***************************
80-89  78 ***************************************
70-79  48 ************************
60-69  30 ***************
50-59  11 ******
40-49  03 **
30-39  03 **
20-29  03 **
```

Check your score against CMS score, let us know if there is a mistake.

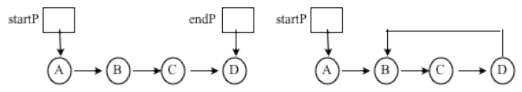
1

Recursion question

/** = "there is a path of male best friends from s to e".

Precondition: s, e not null;
    s and e are male; */

```
public static boolean malePathTo (Person s, Person e) {
    malePathTo(new Vector(), s, e);
}
```



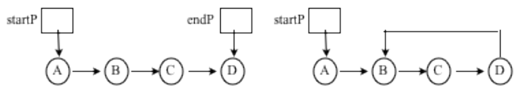
2

Recursion question

/** = "there is a path of male best friends from s to e that does not contain a Person in list ig".

Precondition: s, e, and ig are not null;
    s and e are male;
    s and e are not in ig. */

```
public static boolean malePathTo(Vector<Person> ig,
                                 Person s, Person e)
```



3

Recursion question

/** = "there is path of male best friends from s to e that does not contain a Person in list ig".
Precond: s, e, and ig are not null; s and e are male; s and e are not in ig. */

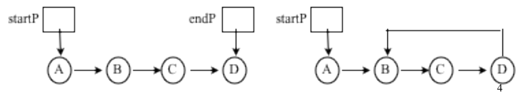
```
public static boolean malePathTo(Vector<Person> ig, Person s, Person e) {
    if (s == e) return true;
    if (s.getMBF() == null) return false;
    ig.add(s);
    if (ig.contains(s.getMBF())) return false;
    return malePathTo(ig, s.getMBF, e);
}
```

base case: path of one node!!!!!

make sure s is not looked at again

Look at the def of the path above

Note that this call satisfies all parts of the precondition of the specification.



4

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.

Two ideas on test cases:
1. Black Box Testing: Develop test cases based on the spec.
2. White Box Testing: Look at the code; develop test cases so that each statement/expression is exercised in at least one test case.

5

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

call stack

```
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)
```

6

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

Executing Java programs outside the DrJava Interactions pane

The Java application

A Java application needs a class with a method main that is defined like this.

```
public class C {
    …
    public static void main(String[] args) {
        …
    }
    …
}
```

To start the application, method main is called.

The parameter, an array of Strings, can be used to pass information into the program.

13

```
public class CLAS {
    …
    public static void main(String[] args) {
        …
    }
    …
}
```

```
> cd
> dir
  (list of files)
> java CLAS
```

Causes method call CLAS.main(null); to be executed

Terminal window

(can type "java CLAS" in DrJava Interactions pane)

14

jar file (Java Archive file)

(like tar file (Tape Archive file))

Look in index of text for info on jar files and how to build them in the command-line window.

Contains (among other things)

(1) .class files

(2) a "manifest", which says which class has method main

Manifest:

A list of passengers or an invoice of cargo for a vehicle (as a ship or plane).

15

Standalone Java programs

Suppose images.jar contains a Java application

It has a class with a static procedure main, and its manifest names the class.

- Execute it by double clicking its icon in a directory.
- Execute it by typing

java -jar images.jar

in a terminal window (or DOS, or command-line window)

16

Inspecting jar files

```
jar   -tf   images.jar
```

List the contents of jar file images.jar

type (list)

File

name of jar file

17

Creating a jar file

1. Navigate to the directory that contains the .class files.
2. Create a text file x.mf that contains one line (with a line-feed):

   Main-class: <name of class>

3. In the directory, type:

```
jar   -cmf   x.mf   app.jar   *.class
```

Create

Manifest

File

name of manifest file

name of file to create

expands to name all the .class files

18