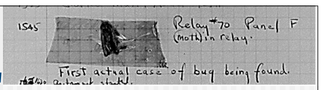
Computer Science 101 S1

"Oh no, not computer bugs again !"

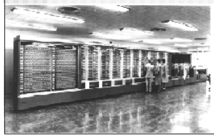
Lecture 20

Contents

The First "Computer Bug"


1545
Relay #70 Panel F (moth) in relay.
First actual case of bug being found.

Moth found by Grace Hopper trapped between points at Relay #70, Panel F, of the Harvard Mark II Aiken Relay Calculator (similar to the Mark I shown below) while it was being tested at Harvard University, 9 September 1945. The operators affixed the moth to the computer log, with the entry: "First actual case of bug being found". They put out the word that they had "debugged" the machine, thus introducing the term "debugging a computer program"



The Mark I was constructed out of switches, relays, rotating shafts, and clutches, and was described as sounding like a "roomful of ladies knitting." The machine contained more than 750,000 components, was 50 feet long, 8 feet tall, and weighed approximately 5 tons!

1. Compile Time messages

This compiler message is telling me that the interpreter cannot find the class, `Slide05`.

```
> javac L20.java
L20.java:30: cannot resolve symbol
symbol   : class NoClass
location: class L20
     NoClass d;
     ^
1 error
```

This compiler message also tells me that the problem code is on line 30 of the **L20** class.

2. Compile Time messages

It may also happen that the name of a public class is different to the name of the file in which the class definition is stored. This compiler message is telling me that the `Slide06` class needs to be stored in a file called `Slide06.java`. (Currently the `Slide06` class is stored in a file called `Slide16.java`.)

```
> javac L20.java
Slide15.java:1: class Slide05 is public,
should be declared in a file named Slide05.java
public class Slide05 {
             ^
1 error
```

3. Compile Time messages

When I compile the following code:

```
String s;
System.out.println(s);              //line 53

Circle st;
System.out.println(st);             //line 56
…
```

I get the error message shown on the next slide. Change the code so that my program compiles.

3. Compile Time messages

```
> javac L20.java
Program.java :53: variable s might not have
been initialized
 System.out.println(s);             //line 53
                    ^
Program.java :56: variable st might not have
been initialized
 System.out.println(st);            //line 56
                    ^
2 errors
```

4. Compile Time messages

When I compile the following code:

```
int i = 3.4;                             //line 71
String s = 34;                           //line 72
System.out.println("i: " + i);
System.out.println("s: " + s);
 …
```

I get the error message shown on the next slide.
Change the code so that my program compiles.

4. Compile Time messages

```
> javac L20.java
Program.java :71: possible loss of precision
found    : double
required: int
    int i = 3.4;
            ^
Program.java :72: incompatible types
found    : int
required: java.lang.String
    String s = 34;
               ^
2 errors
```

5. Compile Time messages

When I compile the following code:

```
String s = (String) 34;                 //line 88
System.out.println("s: " + s);
 …
```

I get the error message shown on the next slide.
Change the code so that my program compiles.

5. Compile Time messages

```
> javac L20.java
Program.java :88: inconvertible types
found    : int
required: java.lang.String
    String s = (String) 34;
                        ^
1 error
```

Reminder about Running Code

For the following slides, two classes are defined, `L20` application:

```
public class L20 {
  public static void main(String[] args) {
      Program p = new Program();
      p.start();
} }
```

and `Program` (containing the `start()` method):

```
public class Program {
  public void start() {
      ...
  }
  private void slide12() {
      ...
  }
}
```

The application class is executed: `> java L20`

6. Run Time Errors

The following code compiles but when I run it:

```
public class Program {
  public start() {
     …..
     slide13();                        //line 11
  }
  private void slide13() {
     String s = null;
     System.out.println(s.length());   //line 96
     …
```

I get the error message shown on the next slide. Change the code so that my program both compiles and runs.

6. Run Time Errors

```
> java L20
Exception in thread "main"
java.lang.NullPointerException
             at Program.slide13(Program.java:96)
               at Program.start(Program.java:11)
                         at L20.main(L20.java:4)
```

Don't forget to use the information given to you about the run-time error by the Java virtual machine.
The method name where the problem occurred is given (and the method name from which this method was called), also the class name and line number.

7. Run Time Errors

The following code compiles but when I run it:

```
public class Program {
  public start() {
     …..
     slide15();                         //line 12
  }
  private void slide15() {
     int i = Integer.parseInt("ABC45"); //line110
     System.out.println("i:  " + i);
     …
```

I get the error message shown on the next slide. Change the code so that the program compiles and runs.

7. Run Time Errors

```
> java L20
Exception in thread "main"
java.lang.NumberFormatException:
For input string: "ABC45" at
java.lang.NumberFormatException.forInputString(
                  NumberFormatException.java:48)
  at java.lang.Integer.parseInt(Integer.java:426)
  at java.lang.Integer.parseInt(Integer.java:476)
              at Program.slide15(Program.java:110)
                at Program.start(Program.java:12)
                         at L20.main(L20.java:4)
```

Notice the method trace (stack trace).

8. Run Time Errors

The following code compiles but when I run it:

```
public class Program {
  public start() {
     …..
     slide17();                       //line 13
  }
  private void slide17() {
     String s = "Hello";
     System.out.println("9th character: " +
                         s.charAt(8));  //line 124
     …
  }
}
```

Change the code so that there is no longer a runtime exception when the `slide17()` method is executed.

8. Run Time Errors

```
> java L20
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException:
String index out of range: 8
     at java.lang.String.charAt(String.java:460)
             at Program.slide17(Program.java:124)
               at Program.start(Program.java:13)
                         at L20.main(L20.java:4)
```

Notice the method trace (stack trace).

Random Characters From a String

Problem: Write some java code which creates and then prints a string of three randomly selected characters from the String "ABCDEF". Every character in your string must be unique.

I write my code (shown on slide 21) and run the program. Sometimes it works :- but sometimes it doesn't :- (Look at the example output on the next slide.)

Random Characters From a String

```
> java L20
Selected characters: EAC
> java L20
Selected characters: BAC
> java L20
Selected characters: FAD
> java L20
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException: String index
out of range: 3
            at java.lang.String.charAt(String.java:460)
                   at Program.slide21(Program.java:147)
                      at Program.start(Program.java:15)
                                at L20.main(L20.java:4)
```

Random Characters From a String

```
public class Program {
 public start() {
      slide21();                                  //line 15
 }
 private void slide21() {
   String chars = "ABCDEF";
   String chosen= "";
   int charNum;
   for(int i=0; i<3; i++) {
     charNum = (int)(Math.random() * chars.length());
     chars = chars.substring(0, charNum)
              + chars.substring(charNum + 1);  //line 146
     chosen = chosen + chars.charAt(charNum); //line 147
   }
   System.out.println("Selected characters: " + chosen);
 }
}
```

9. Use `System.out.println()`

Use `System.out.println()` statements to print the values of the relevant variables. This way you will be able to see what variable values are stored in memory and how these variable values change as your code is executed. On the next slide you can see that I have inserted three `System.out.println()` statements inside the loop.

On slide 25 you can see some sample output when the java code from slide 23 is executed.

Can you work out where the problem is in the code and correct it?

9. Use `System.out.println()`

```
String chars = "ABCDEF", chosen = ""; int charNum;
for(int i=0; i<3; i++) {
   charNum = (int) (Math.random() * chars.length());
   System.out.println("i: " + i + ", chosen " + chosen
                          + ", charNum: " + charNum);
   System.out.println("chars before substring: " + chars);
   chars = allChars.substring(0,charNum)
                 + chars.substring(charNum+1); //line146
   System.out.println("chars after substring: " + chars);
   chosen = chosen + chars.charAt(charNum);   //line 148
   System.out.println();
}
System.out.println("Selected characters: " + chosen);
```

9. Use `System.out.println()`

```
> java L20
i: 0, chosen , charNum: 2
chars before substring: ABCDEF
chars after substring: ABDEF
i: 1, chosen D, charNum: 0
chars before substring: ABDEF
chars after substring: BDEF
i: 2, chosen DB, charNum: 3
chars before substring: BDEF
chars after substring: BDE
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException: String index
out of range: 3
             at java.lang.String.charAt(String.java:460)
                    at Program.slide23(Program.java:148)
                       at Program.start(Program.java:15)
                                 at L20.main(L20.java:4)
```

10. Use `System.out.println()`

Problem: The user is given a random number which they can choose to increase or decrease by a random amount between 1 and 5. The user's aim is to end up with a number as close as possible to 20. The user can use up to 5 attempts to change the number.

Debugging: use `System.out.println()` to print out messages saying which part of the code has been reached.

The following java code has a problem. From the output shown on slide 27 can you find the problem?

```
boolean hasFinished = false;
int userNum = (int) (Math.random() * 10 + 15);
int compNum = (int) (Math.random() * 5 + 16);
String userChoice;
int numTries = 0;
```

... cont.

10. Use `System.out.println()`

```
while(hasFinished == false) {
   numTries++;
   System.out.println("Your number is: " + userNum);
   System.out.print("TYPE EITHER:  i (increase), d
                              (decrease) OR s (stop) ");
   userChoice = Keyboard.readInput();
   if(userChoice.equals("d")) {
      System.out.println("IN THE d SECTION ");
      userNum = userNum - (int)(Math.random() * 5 + 1);
   }
   if(userChoice.equals("s") || numTries>4) {
      System.out.println("IN THE s SECTION ");
      hasFinished = true;
   } else {
      System.out.println("IN THE else SECTION ");
      userNum = userNum + (int)(Math.random() * 5 + 1);
   }
   System.out.println();            }
```

10. Use `System.out.println()`

```
Your number is: 16
TYPE EITHER:  i (increase), d (decrease) OR s (stop) i
IN THE else SECTION
Your number is: 20
TYPE EITHER:  i (increase), d (decrease) OR s (stop) i
IN THE else SECTION
Your number is: 23
TYPE EITHER:  i (increase), d (decrease) OR s (stop) d
IN THE d SECTION
IN THE else SECTION
Your number is: 27
TYPE EITHER:  i (increase), d (decrease) OR s (stop) d
IN THE d SECTION
IN THE s SECTION
Number of Tries: 5  Your number: 25, Computer number: 17
```

This number should have decreased not increased???

11. Comment out sections of code

If you can't find a problem, comment out the section of code which includes the problem area:

```
 ………..
 for(int i=0; i<5; i++) {
/*
   charNum = (int)(Math.random()*codes.length());
   codes = codes.substring(0,charNum)
      + codes.substring(charNum+1);
   chosen += codes.charAt(charNum);
*/

 }
  ……….
```

11. Comment out sections of code

Compile and execute the code and keep reducing or increasing the number of lines commented out until you can narrow the problem down.

```
 ………..
 for(int i=0; i<5; i++) {
   charNum = (int)(Math.random() * codes.length());

/*
   codes = codes.substring(0, charNum)
                    + codes.substring(charNum+1);
   chosen += codes.charAt(charNum);
*/

 }
  ……….
```

12. Use Methods

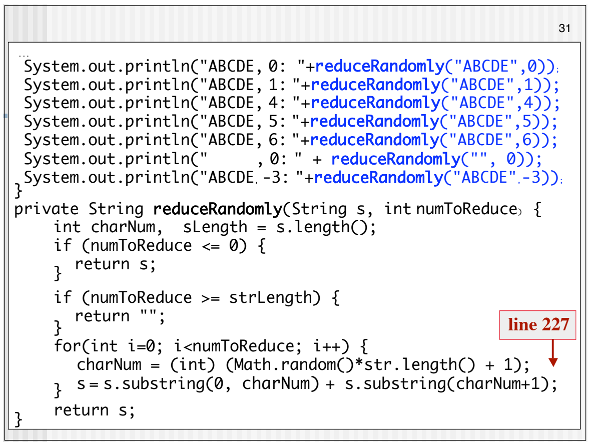
The `reduceRandomly()` method is passed two parameters:
- `s` - the string from which characters are to be randomly removed,

and
- `numToReduce` - the number of characters to be removed from the string.

The method should randomly remove the required number of characters and return the resulting string. The code shown on slide 31 has a bug.

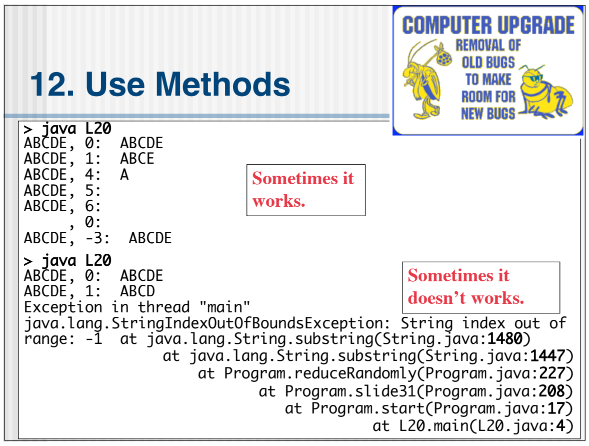
To test the `reduceRandomly()` method I can call this method many times with differing parameter values as shown on slide 31.

The ouput shown on slide 32 may help you detect the problem.

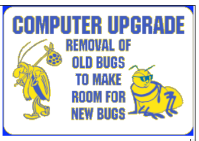


```
...
System.out.println("ABCDE, 0: "+reduceRandomly("ABCDE",0));
System.out.println("ABCDE, 1: "+reduceRandomly("ABCDE",1));
System.out.println("ABCDE, 4: "+reduceRandomly("ABCDE",4));
System.out.println("ABCDE, 5: "+reduceRandomly("ABCDE",5));
System.out.println("ABCDE, 6: "+reduceRandomly("ABCDE",6));
System.out.println("     , 0: " + reduceRandomly("", 0));
System.out.println("ABCDE, -3: "+reduceRandomly("ABCDE",-3));
}
private String reduceRandomly(String s, int numToReduce) {
    int charNum,  sLength = s.length();
    if (numToReduce <= 0) {
      return s;
    }
    if (numToReduce >= strLength) {
      return "";
    }
    for(int i=0; i<numToReduce; i++) {
      charNum = (int) (Math.random()*str.length() + 1);
      s = s.substring(0, charNum) + s.substring(charNum+1);
    }
    return s;
}
```

line 227



12. Use Methods

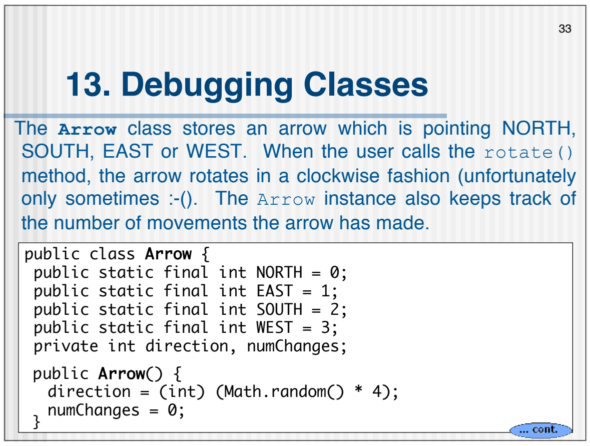


```
> java L20
ABCDE, 0:  ABCDE
ABCDE, 1:  ABCE
ABCDE, 4:  A
ABCDE, 5:
ABCDE, 6:
     , 0:
ABCDE, -3:  ABCDE
```

Sometimes it works.

```
> java L20
ABCDE, 0:  ABCDE
ABCDE, 1:  ABCD
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException: String index out of
range: -1  at java.lang.String.substring(String.java:1480)
            at java.lang.String.substring(String.java:1447)
            at Program.reduceRandomly(Program.java:227)
            at Program.slide31(Program.java:208)
            at Program.start(Program.java:17)
            at L20.main(L20.java:4)
```

Sometimes it doesn't works.



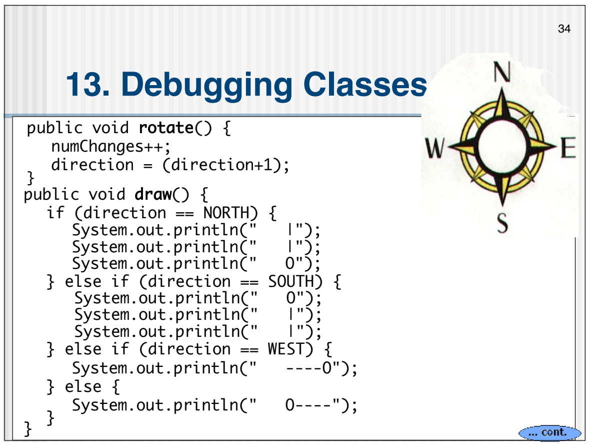
13. Debugging Classes

The **Arrow** class stores an arrow which is pointing NORTH, SOUTH, EAST or WEST. When the user calls the rotate() method, the arrow rotates in a clockwise fashion (unfortunately only sometimes :-(). The Arrow instance also keeps track of the number of movements the arrow has made.

```
public class Arrow {
  public static final int NORTH = 0;
  public static final int EAST = 1;
  public static final int SOUTH = 2;
  public static final int WEST = 3;
  private int direction, numChanges;

  public Arrow() {
    direction = (int) (Math.random() * 4);
    numChanges = 0;
  }
```

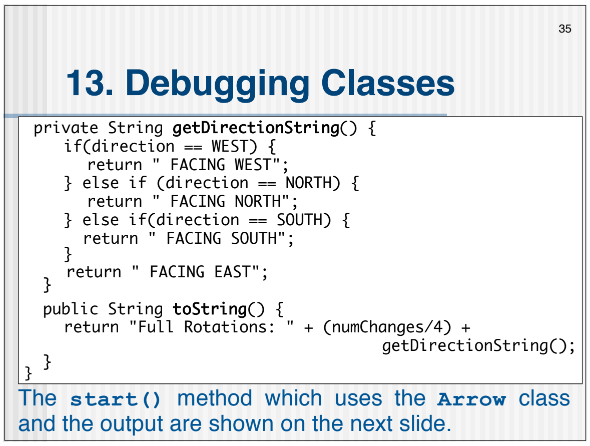
... cont.



13. Debugging Classes

```
public void rotate() {
   numChanges++;
   direction = (direction+1);
}
public void draw() {
   if (direction == NORTH) {
      System.out.println("   |");
      System.out.println("   |");
      System.out.println("   0");
   } else if (direction == SOUTH) {
      System.out.println("   0");
      System.out.println("   |");
      System.out.println("   |");
   } else if (direction == WEST) {
      System.out.println("   ----0");
   } else {
      System.out.println("   0----");
   }
}
```

... cont.

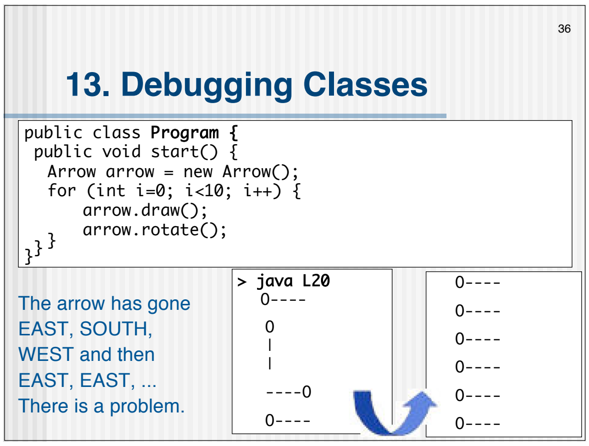


13. Debugging Classes

```
  private String getDirectionString() {
      if(direction == WEST) {
        return " FACING WEST";
      } else if (direction == NORTH) {
        return " FACING NORTH";
      } else if(direction == SOUTH) {
        return " FACING SOUTH";
      }
      return " FACING EAST";
   }

   public String toString() {
      return "Full Rotations: " + (numChanges/4) +
                                        getDirectionString();
   }
}
```

The **start()** method which uses the **Arrow** class and the output are shown on the next slide.



13. Debugging Classes

```
public class Program {
  public void start() {
    Arrow arrow = new Arrow();
    for (int i=0; i<10; i++) {
        arrow.draw();
        arrow.rotate();
}}}
```

The arrow has gone EAST, SOUTH, WEST and then EAST, EAST, ... There is a problem.

```
> java L20
  0----

  0
  |
  |

  ----0

  0----
```

```
0----

0----

0----

0----

0----

0----
```

13. Debugging Classes

When debugging it is helpful to include an instance method which prints the current values of ALL the instance variables. See the `printDetails()` method:

```
public class Arrow {
  public static final int NORTH = 0;
    ....
  public void printDetails() {
     System.out.println("Direction: " + direction +
        getDirectionString() + ", numChanges: " + numChanges);
  }
}
```

```
public class Program {
  public void start() {
     Arrow arrow = new Arrow();
     for (int i=0; i<10; i++) {
         arrow.printDetails();
         arrow.rotate();
} } }
```

13. Debugging Classes

This is the output. Can you spot the problem?

```
> java L20
Direction: 0 FACING   NORTH, numChanges: 0
Direction: 1 FACING   EAST, numChanges: 1
Direction: 2 FACING   SOUTH, numChanges: 2
Direction: 3 FACING   WEST, numChanges: 3
Direction: 4 FACING   EAST, numChanges: 4
Direction: 5 FACING   EAST, numChanges: 5
Direction: 6 FACING   EAST, numChanges: 6
Direction: 7 FACING   EAST, numChanges: 7
Direction: 8 FACING   EAST, numChanges: 8
Direction: 9 FACING   EAST, numChanges: 9
```

What you need to know

Be able to read and understand some of the compile error messages

Debug using System.out.println() statements to check what your code is storing.

Debug using System.out.println() statements to check which parts of your code is being executed.

Use methods. (It is easy to test methods.)