Testing

Learning Outcomes

- Create a test plan
- Identify special characteristics of testing when considering data files, if statements, loops and methods

Testing

“Debugging can reveal the presence of bugs, but never their absence.”

E.W. Djikstra

How to Do Testing?

- Start with easy examples
- Use multiple sets of data
- Create a *test plan*
  - Reasons
  - Specific inputs
  - Expected outputs
  - Actual outputs
- Black box vs. glass box testing
- Data coverage vs. code coverage

Example

Write a method to determine if a number is even.

boolean isEven (int number)

Reason	Specific Input(s)	Expected Output	Actual Output
A one-digit odd number	1	False	
A one-digit even number	2	True	

Test Plan Example

Reason	Specific Input(s)	Expected Output	Actual Output
A single digit odd number	1	False	
A single digit even number	2	True	
Two-digit odd number	11	False	
Two-digit even number	10	True	
Negative even number	-2	True	
Negative odd number	-1	False	
Positive big number	100003	False	
Negative big number	-900302	True	
Zero	0	True	

File Input/Output

- Test a missing file
- Test an empty input file
- Test with a “full” file

If statements

```
if (x > 4)
   // do this
else
   // do that
```

Loops

```
while (x < 14)
{
    :
}
```

- Check for infinite loop
- Check initialization
- Check order of statements in body of loop
- Check “off by one” error
- Check for correct behaviour if no loop execution

Methods

$$\begin{tikzcd}
 & & F1 \arrow[dll] \arrow[d] \arrow[drr] & & \\
F2 & & \overset{\blacklozenge}{\underset{\blacklozenge\ \blacklozenge}{F3}} \arrow[dl] \arrow[dr] & & F4 \\
 & F5 & & F6 \arrow[dr] & \\
 & & & & F7
\end{tikzcd}$$

Testing and Debugging Techniques

- Insert output statements
- Comment out chunks of code
- Hand trace the code
- Use stubs and drivers
- Use a debugger

Test Plan and Program

- Write a test plan and a program for the following problem.
  - Accept a string from the user and determine if there are twice as many a’s as e’s in it.