Assertsions and Loop Invariants

Handout: “Assertions and Loop Invariants”

I am indebted to my colleague, John David Stone, for assistance with this handout. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. To view a copy of this license, visit https://creativecommons.org/licenses/by-sa/4.0/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.
Assert-statement

• Test precondition
• Interrupt execution if precondition is not met
• Places that assert-statement is used:
  – Beginning of method
  – Before a loop
  – Before returning statement
Assert-statement (cont.)

```java
assert <Boolean expression>;
  • If <Boolean expression> is false, AssertionError is thrown.

• assert <Boolean expression> <expression>;

• Enable assertions
  – Command line: java -ea Echo
  – Eclipse: Run Configurations -> Arguments -> VM arguments
```
Assert-statement (cont.)

- When to use assert-statement?
  - Untested preconditions and postconditions
  - Develop, test, and maintain a program
Loop Invariant

- Use loop invariant to check the correctness of the body of loops.
- Formulate loop invariant statements for complex loops.
- Identify the result of a loop and transform it to a loop invariant.
Loop Invariant

• A condition that is true
  – Before execution of a loop
  – At the end of every iteration of the loop
  – After the execution of the loop

```cpp
int factorial( int n ) {
    int ret;
    for ( int i = 1; i <= n; i++ ) {
        ret *= i;
    }
    return ret;
}
```

Loop invariant: ret contains the product of numbers from 1 to k, where k is the count of numbers that we have seen so far.