Reading 13: A First Look at Java

# Exercise 1: Summarize

Java is a well-structured, object-oriented language that promotes code reuse and maintainability through inheritance, polymorphism, and encapsulation, leading to strong efficiency, security, and reliability.

# Exercise 2: Demonstrate & Explain

ConsCell:

public static void testConsCell() {

ConsCell a = null;

ConsCell b = new ConsCell(2, a);

ConsCell c = new ConsCell(1, b);

assert a == null : "FAIL: a should be an empty list";

assert b.getHead() == 2 : "FAIL: b head should be 2";

assert b.getTail() == null : "FAIL: b tail should be null";

assert c.getHead() == 1 : "FAIL: c head should be 1";

assert c.getTail().getHead() == 2 : "FAIL: c tail should start with 1";

assert c.getTail().getTail() == null : "FAIL: c tail should end with null";

}

public static void main(String[] args) {

testConsCell();

}

IntList:

public static void testIntList() {

IntList list = new IntList(null);

assert list.length() == 0 : "FAIL: list ln should be 0";

list = list.cons(1); // push

assert list.length() == 1 : "FAIL: list should be 1 element";

list = list.cons(2); // push

assert list.length() == 2 : "FAIL: list should be 2 elements";

ConsCell cell = list.start;

assert cell.getHead() == 2 : "FAIL: list[0] should be 2";

assert cell.getTail().getHead() == 1 : "FAIL: list[1] should be 1";

assert cell.getTail().getTail() == null : "FAIL: list[2] should be null";

}

public static void main(String[] args) {

testIntList();

}

Driver:

public static void main(String[] args) {

try {

testConsCell();

} catch (AssertionError e) {

assert false : "FAIL: testConsCell failed";

}

try {

testIntList();

} catch (AssertionError e) {

assert false : "FAIL: testIntList failed";

}

}

# Exercise 3: Inquire

Can you explain the difference between a reference and a pointer in Java, and why Java is often said to be ‘like C++ without pointers’?