References: contains the address of the object

Primitive: contains the value itself

Aliases: two or more references that refer to the same object are called aliases of each other

One object can accessed using multiple reference variables.

int x = 5;

if (x = 4) {

}

else if (x = 5) {

}

else {

}

switch (x) {

case 1:

case 2:

default:

break;

}

x = (x == 5 ? 3 : 4); // conditional operator

while (x > 0) {

--x;

}

do {

--x;

} while (x > 0);

for (int i = 0; i < 5; ++i) {

x += 2;

}

Interfaces: a collection of abstract methods and constants.

Abstract method is a method header without a method body.

All methods in an interface are abstract.

An abstract method can be declared using modifier abstract.

An interface is used to establish a set of methods that a class will implement.

implement is a reserved word for a class to implement an interface.

extends is a reserved word for a class to inherit from another class.

Comparable interface contains one abstract method called compareTo, which is used to compare two objects.

if (obj1.compareTo(obj2) > 0) {

System.out.println(“obj1 is greater than obj2.”);

else if (obj1.compareTo(obj2) < 0) {

System.out.println(“obj1 is less than obj2.”);

else {

System.out.println(“The two objects are equal.”);

}

Dependency: RollingDie use a die.

Composition: PairOfDice has two dice.

Aggregation: Car has a passenger.

Inheritance: LoadedDie is a die.

Primitive type: pass by value

Reference type: pass by reference

int[] array = new int[5];

int[][] table = new int[4]; // 4 rows

for (int i = 0; i < 4; ++i) {

table[i] = new int[4]; // 4 columns

}

array vs ArrayList

array can hold both primitive and reference types.

ArrayList can only hold reference types.

array has fixed size.

ArrayList has dynamic size.

Initializer lists

int[] array = { 1, 2, 3, 4, 5 };

Bounds checking

array.length holds the number of elements, not the largest index.

arrayList.size() holds the number of elements, not the largest index.

Test cases: a set of input and user actions, coupled with the expected results.

Regression testing: running previous test suites to ensure new errors haven’t been introduced.

Black box testing: test cases are developed without considering the internal logic. They are based on the input and expected output. Two input values in the same equivalence category would produce similar results. Therefore, a good test suite will cove all equivalence categories and focus on the boundaries between categories.

White box testing: developers focus on the internal structure of the code. The goal is to ensure that every path through the code is tested.

Integration test: the process of testing software components that are made up of other interacting components. Stresses the communication between components rather than the functionality of individual components.

Regression test: running previous test cases after a change is made to a program to help ensure that the change did not introduce an error.

Review: a meeting in which several people collectively evaluate an artifact.

Walkthrough: a review that steps carefully through a document, evaluating each section.

Defect: the goal of testing is to discover these.

Test cases: a description of the input and corresponding expected output of a code unit being tested.

Test suite: a set of test cases that covers various aspects of a system.

Black-box: with this testing approach, test cases are based solely on requirement specifications.

White-box: with this testing approach, test cases are based on the internal workings of the program.