Rio Salado College

Key Terms and Concepts – Lesson 3

CIS163AA – Java Programming – Level 1

Term	Definition
Abstract data type	An **abstract data type** is a type whose implementation is hidden and access through its public methods.
Abstraction	**Abstraction** is the programming feature that allows you to use a method name to encapsulate a series of statements.
Access modifier	**Access modifier** is sometimes used as another term for access specifier.
Accessor methods	**Accessor methods** retrieve values.
Actual parameters	**Actual parameters** are the arguments in a method call.
Arguments	**Arguments** are data items sent to methods in a method call.
Black box	A **black box** is a device you can use without understanding how it works.
Class client class user	A **class client** or **class user** is an application or class that instantiates objects of another prewritten class.
Classes	**Classes** can be extended used as a basis for any other class.
Client method	A **client method** is a method that calls another.
Constructor	A **constructor** is a method that establishes an object.
Dead code	**Dead code** is a set of statements that are logically unreachable.
Declaration	A **declaration** is another name for a method header.
Default constructor	A **default constructor** is one that is created automatically by the Java compiler.
Fields	**Fields** are data variables declared in a class outside of any method.
Formal parameters	**Formal parameters** are the variables in a method declaration that accept the values from actual parameters.
Fully qualified identifier	A **fully qualified identifier** includes a class name and a dot before the identifier.
Implementation hiding	**Implementation hiding** is a principle of object-oriented programming that describes the encapsulation of method details within a class.

Term	Definition
Information hiding	**Information hiding** is the object-oriented programming principle used when creating private access for data fields; a class’s private data can be changed or manipulated only by a class’s own methods and not by methods that belong to other classes.
Instance variables	The **instance variables** of a class are its data components.
Instantiation	An **instantiation** of a class is an object; in other words, it is one tangible example of a class.
Interface	The **interface** to a method includes the method’s return type, name, and arguments. It is the part that a client sees and uses.
Invoke call	When you **invoke** or **call** a method, you execute it.
Is-a relationship	An **is-a relationship** is the relationship between an object and the class of which it is a member.
Local variable	A **local variable** is known only within the boundaries of a method.
Method	A **method** is a program module that contains a series of statements that carry out a task.
Method call	The calling method makes a **method call** that invokes the called method.
Method body	A **method body** is the set of statements between curly braces that follow the header and that carry out the method’s actions.
Method’s type	A **method’s type** is its return type.
Method's signature	A **method's signature** is the combination of the method name and the number, types, and order of arguments. ’
Mutator methods	**Mutator methods** set values.
New operator	The **`new` operator** allocates the memory needed to hold an object.
Nonstatic methods	**Nonstatic methods**, those methods used with object instantiations, are called instance methods.
Parameters	**Parameters** are the data items received by a method.
Primary key	A **primary key** is a unique identifier for data within a database.

Term	Definition
Private access	Assigning **private access** to a field means that no other classes can access the field's values, and only methods of the same class are allowed to set, get, or otherwise use private variables.
Programmer-defined data type	A **programmer-defined data type** is one that is created by a programmer and not built into the language.
Reference to an object	A **reference to an object** is the name for a memory address where the object is held.
Return a value	To **return a value** is to send the value from a called method back to the calling method.
Return statement	A **return statement** ends a method and frequently sends a value from a called method back to the calling method.
Return type	A **return type** indicates the type of data that, upon completion of the method, is sent back to its calling method.
Stub	A **stub** is a method that contains no statements; programmers create stubs as temporary placeholders during the program development process.
Unreachable statements	**Unreachable statements** are those that cannot be executed because the logical path can never encounter them; an unreachable statement causes a compiler error.

Private access	Assigning **private access** to a field means that no other classes can access the field's values, and only methods of the same class are allowed to set, get, or otherwise use private variables.
Programmer-defined data type	A **programmer-defined data type** is one that is created by a programmer and not built into the language.
Reference to an object	A **reference to an object** is the name for a memory address where the object is held.
Return a value	To **return a value** is to send the value from a called method back to the calling method.
Return statement	A **return statement** ends a method and frequently sends a value from a called method back to the calling method.
Return type	A **return type** indicates the type of data that, upon completion of the method, is sent back to its calling method.
Stub	A **stub** is a method that contains no statements; programmers create stubs as temporary placeholders during the program development process.
Unreachable statements	**Unreachable statements** are those that cannot be executed because the logical path can never encounter them; an unreachable statement causes a compiler error.

Note: Please see key terms in the textbooks for examples of some of the terms.