Keyword Terminology

- **catch**: defines how a particular kind of exception is handled
- **extends**: a class is derived from an existing class
- **final**: the value of the variable cannot be changed
- **finally**: always executes after either a try or catch
- **implements**: a class must provide method implementations to an interface
- **import**: include classes from other libraries
- **interface**: a class containing a collection of constants and abstract methods
- **new**: call a class’s constructor
- **null**: a reference to no object
- **private**: visible within the same class
Keyword Terminology

- **protected**: visible within the same class or subclass
- **public**: visible from within and outside a class
- **super**: used in a subclass to refer to the parent class
- **this**: a reference to the current object (self)
- **throw**: creates an exception
- **try**: contains a group of statements that may thrown an exception
Concept Terminology

- **abstract data type**: a data type whose values and operations are not inherently defined within a programming language
- **abstraction**: (a.k.a. information hiding) making implementation details inaccessible; hiding “the right amount” of complexity
- **abstract class**: a generic concept in a class hierarchy; cannot be instantiated and usually contains one or more abstract methods
- **abstract method**: a method that does not have an implementation
- **actual parameter**: parameters sent into the method
- **alias**: two variables that reference the same object
Concept Terminology

- **collection**: an object that gathers and organizes other objects
- **constructor**: special method that shares the same name as the class and initializes an object
- **data structure**: the collection of programming constructs used to implement an abstract data type
- **dynamic binding**: (a.k.a. late binding) binding a method invocation to a method definition at run time
- **encapsulation**: variables contained within an object should only be accessible from within that object
- **error**: generally represents an unrecoverable situation and should not be caught
- **exception**: an object that defines an unusual or erroneous situation
Concept Terminology

• **exception propagation**: when an exception isn’t handled immediately, control returns to the calling method

• **formal parameter**: parameters located inside the method

• **inheritance**: creating a new class that is based on an existing class

• **instance variable**: variable declared outside of methods in a class

• **interface**: the public methods through which we can interact with an object

• **is-a relationship**: the derived class should be a more specific version of the superclass
Concept Terminology

- **linked structure**: a data structure that uses object reference variables to create links between objects
- **method overloading**: multiple methods with the same name but different method signatures
- **method signature**: the method name, number of parameters, types of those parameters, and ordering of those types of parameters
- **modularity**: dividing a large program into small components; each module should perform one well-defined task
- **object reference**: a variable whose value is a memory location; the memory location specifies where the object is located in memory
Concept Terminology

- **polymorphism**: a reference variable that can refer to different types of objects at different points in time
- **scope**: the part of a program in which a valid reference to a variable can be made
- **stack**: a linear collection whose elements are added and removed from the top
- **static method**: (a.k.a. class method) a method that can be called without first needing an object to be created; it is called from the class instead of the object, e.g. `Math.Random()`
- **static variable**: (a.k.a class variable) a variable that is shared across all instances of a class; every object created with that class type shares the same static variables
Concept Terminology

- **subclass**: (a.k.a. child class) the class that is based on an existing class (inherits from a superclass)
- **superclass**: (a.k.a. parent class, base class) the class that is used to derive a new class