

Student Name _____

- Designing a class consists of identifying its:
 - behavior: the operations that can be applied to a class object
 - attributes: the data that must be stored to characterize a class object
- Design a class's behavior (operations) independently of implementation details of its attributes.
- Objects should carry within themselves the ability to perform their operations.
- Use an external perspective when you are *using* a class; use an internal perspective when you are *building* a class.
- A class's behaviors should be identified before its attributes because:
 - it is often not obvious what the attributes should be.
 - knowing the class behavior can help with identifying the attributes.
 - behaviors should be independent of any particular details of how the attributes are implemented.
- To identify a class's attributes, go through the list of operations and identify what information each requires. Information that is needed by several different operations is probably an attribute.
- For attributes that change over time or that differ from one object to another, define variables to store their values. Attributes that are the same for all objects and do not change can be represented as constants. Attributes determine the state of an object at a particular time.
- It is good programming practice to hide all attribute variables of a class by making them private.
- Objects share static (or class) constants and variables. Objects have their own copies of non - static constants and variables.
- Class operations are usually implemented using instance methods, of which five common categories are: constructors, accessors, mutators, converters, and utilities.
- When implementing class operations with methods, it is a good practice to define methods that support output early.
- If a class has a `toString()` method, the existing `print()` and `println()` methods can then be used to display objects of that class, because they send the `toString()` message to any object they are asked to display.
- The compiler calls a constructor whenever a class object is defined using the `new` operator. For this reason, when building a class, always provide one or more constructors to initialize the attribute variables of the class. Otherwise, the compiler will generate a constructor to initialize them with default values.

Student Name _____

- A default constructor is invoked when a declaration of an object provides no initial values for the attribute variables.
- An explicit - value constructor has parameters that are used to initialize some or all of an object's attribute variables.
- `static` (or class) methods and instance (or object) methods have the following differences:
Static methods are declared using the keyword `static`; instance methods are not.
Instance methods may access both static and non - static variables, constants, and methods.
Static methods may access only static variables, static constants, and static methods.
Objects share the same copy of static variables, constants, and methods, but they have their own distinct copies of non - static variables, constants and methods.
- A method that must access attribute variables must be defined as an instance method. Otherwise, it should be defined as a static method, and the information it needs should be passed to it via parameters or declared using static variables and constants.
- Since constants represent values that do not change, they should be declared as static constants.
- Explicit - value constructors are useful to construct objects that must be returned by a method.
- Accessor methods can be used to retrieve, but not modify, the value of an object's attribute variables.
- Mutator methods can change the values of an object's attribute variables. They should ensure that the modified object still satisfies the class invariant.
- The keyword `null` should be used to initialize reference - type variables that will immediately be assigned new values.
- Factoring out code that is common to several operations and encapsulating it in a separate method can result in methods that are simpler and, therefore, easier to write and understand.
- The alias problem arises when a copy operation does not produce a completely distinct copy because the address in a reference-type variable is copied rather than the object it refers to.
- A common arrangement of items in a class is as follows:
 - class constants first, so they can be quickly found
 - the methods next, so it is easy to find the class' operations; they make up the class' interface
 - attribute variables last, so the user need not see them.
- Painting a Swing component must always be done with a method named `paintComponent()`.