When should accessor methods be used?

1. **Encapsulation and Control Over Data Access**: When you want to encapsulate the internal state of an object and control how other classes interact with it, getter and setter methods are beneficial. They allow you to restrict direct access to fields and add validation or transformation when setting values. This is particularly useful when setting constraints (like a setAge() method that ensures age is within a valid range).
2. **Consistent Interface**: If an object's internal representation might change over time, getters and setters can provide a stable interface. For example, if you change the structure of data within a class but still want other classes to interact with it the same way, getter and setter methods help maintain a consistent external interface.
3. **Data Binding and Framework Compatibility**: Many frameworks rely on getter and setter methods for data binding and property introspection. For instance, in JavaBeans or many GUI frameworks, these methods are essential for automatic property updates and are often required for tools that need to read or update properties dynamically.
4. **Modularity and Loose Coupling**: If other parts of your codebase rely on specific attributes, getter and setter methods allow those parts to access or modify data without directly depending on the class's internal variables. This decouples dependencies, which is especially useful if you expect to make structural changes in the future.
5. **Logging or Tracing Access**: Getters and setters can also add a level of monitoring, such as logging each time a field is accessed or modified. This is especially valuable for debugging or tracking sensitive data, as you can track changes to the data throughout the program's runtime.

In summary, getters and setters should be used to control data access, maintain consistency, enhance modularity, and comply with framework requirements. However, to avoid overly exposing an object's internal data and to keep code cleaner and more secure, these methods should be implemented only when they serve a purpose aligned with good OO design practices.