**Polymorphism**

Polymorphism is the ability of a code (object, method) to take on different forms or exhibit different behavior depending on the context. In Polymorphism objects of different types can be treated as objects of a common type.   
  
Polymorphism gives room for flexibility and code reusability. It allows the writing of more generic and adaptable code by designing classes and methods that can work with a variety of data types. This makes the code more extensible and easier to maintain because you can add new classes without modifying existing code.

Polymorphism application of polymorphism is in the use of Abstract classes. In abstract classes, methods which are designated as abstract are written without implementation, by this, every derived class must implement that/those methods designated in the parent class.

This can be seen in the Eternal Quest program thus:

public abstract class Goal

{  
 public abstract RecordEvent()  
}

In the above example, the RecordEvent is designated an abstract method, and every derived class must include the implementation of that method. With this you could have different derived classes with the same RecordEvent method but different implementations. This is very similar to the virtual method which uses the word ‘virtual’ to designate a method inside of a class that its implementation in a derived class can be changed while using the keyword ‘override’ to designate the method whose implementation is being overridden.

Also, polymorphism allows the use of interfaces which provides only abstract methods that must be in a derived class.