Brandon Cox, Tyler Nuanes, Austin Uphus
September 14, 2011

Inheritance and Abstract Classes

In order to save time, support reliability, and encourage efficiency, the Java programming language allows for inheritance hierarchies and abstract classes. Whereas inheritance empowers programmers to establish *is-a* relationships between more general and more specific classes that share common methods, abstract classes restrict some of the features of inheritance in cases where the "parent" class should never be instantiated.

Recognizing that more specific classes are often related to more general classes by their methods, inheritance allows more specific subclasses to override or use existing methods from more general superclasses while adding their own constructors, fields, and methods for specialized purposes. Using code that has previously been tested cuts down on errors, time, and overall length of code, allowing programmers to code more efficiently. Additionally, creating a subclass is simple, only requiring one to type `extends Superclass` in the declaration for the subclass; however, Java does not allow subclasses to inherit multiple superclasses at once, so programmers must ensure they use the proper superclass when implementing a subclass.

In contrast to standard superclasses, abstract classes do not implement all of their methods and cannot be instantiated into objects, often because each subclass related to an abstract class implements its shared methods in a different way, making it difficult to create general methods uniting all subclasses. By forcing programmers to implement some of the methods themselves, abstract classes cut down on error; additionally, by creating a common structure, they can save programmers time. Abstract classes and methods are declared with the reserved word `abstract`. Such abstract methods do not contain any code and essentially do nothing; they only provide an organizational structure. In addition, abstract classes can include instance variables, concrete methods, and concrete constructors, unlike interfaces.

Inheritance and abstract classes are most useful because they effectively organize code, increasing programmer literacy. Without these tools, code would be more convoluted, and simple relationships between classes would be harder to visualize.