David Bradshaw

Abstract Data Types

– Approach with Object-Oriented Programming –

Concepts of Programming Languages

Spring 2011

CSCI 3300

Introduction

What is Abstract Data Types?

There is no exact definition that describes what Abstract Data Types are. Many books and references give vague concepts to what they are, but none can give an exact definition. It soon became obvious that there was no exact definition; the only item that gave a form of a definition was the Encyclopædia Britannica, which simply gave the theory that they are for large scale programs, and a manner of hiding internal details of data structures and their operations ("Abstract Data Types"). This was not what was needed for explaining Abstract Data Types to the standard user.

The first step in deciphering what Abstract Data Types are is to break the term up into the basic words that make up this concept. The first word would be 'abstract'; 'abstract' is simply anything that is conceptual or intangible. The next word is 'data'; 'data' is obviously any piece of information that is present, more importantly in this case, information presented to the computer. This leaves our last word 'type'; 'type' is a vague word, which usually shows what area something belongs in (Dale, 2-3). When you put these three separate concepts together you get a conceptual piece of data that belongs to a certain category. This is getting closer to the true theory behind Abstract Data Types. From this, the true meaning of Abstract Data Types are coming to light. It is simply the alterations of conceptual data, or data that is not known or understood, that are altered based on their category or type.

Data Abstraction

The basic concept behind Abstract Data Types is the use of abstraction. Abstraction is seen as creating a vague way of specifying types, or categories, of any general concept (Sebesta, 470). The use of abstraction is used in many science and business fields. For example, in biology, scientist use an abstract view of certain categories of living things, called Taxonomy, and then place all of the animals into their rightful category, and specify additional characteristics present in the specific animal. This is generally the same idea behind abstraction in computer science. Computer scientists take a higher level of data type, like an integer or an array, and create classes for each abstraction that can be referred when needed. The programmer can than declare what type of data is being used and the computer will automatically apply the rules and alterations needed for use in the program. Data abstraction is the basic principle behind Abstract Data Types in computer science.

Abstract Data Types Purpose in Computer Science

Every day, programs are written and programmers use algorithms to compute data and yield results to the console. However, many do not understand the basic theory behind the change in data and how alterations, like multiplication and addition, occur. The theory behind Abstract Data Types is simple, in the awareness of that it makes tasks easier by allowing programmers to focus on what is needed within their programs and not worry about the higher level ideology; like making the computer understand that an integer is a number of some sort. When a programmer declares that the data in question is an integer the programming language allows the programmer to use integer specific items to alter the data (Sebesta, 470-471). If this was not possible, the programmer would have to focus on making the computer realize that an integer is being altered every time an alteration is created for the program. This could lead to very complex and long programs that are hard to read and take way too much time to process by the computer.

Abstract Data Types are used in many computer languages. Abstract Data Types are used in most typed languages; which are languages that use standard words and then are compiled into computer code. This includes many well-known languages like C based languages; C# and C++, Java, web based languages; Ruby and Rails, and older languages; Ada and SIMULA 67 (Mitchell, 1). Essentially languages that are compiled into code use Abstract Data Types since they allow the programmer to reference a piece of data, like a number or character from the ASCII set, and make alterations and procedures onto the data being referenced.

Abstract Data Types

Abstract Data Types Specifications for Programming

As mentioned earlier, many languages, mostly typed languages use Abstract Data Types. This is very true for Java, which is used throughout the paper for examples. C++, Ada, C#, Objective C, and many more use the theory as well. It is nearly impossible to show an example of how this is portrayed in actual code. However, presented below is a snippet from a Java program that computes information on geometric objects, more importantly, for the example, on the triangle object. The snippet of code has been edited and condensed for the example, most code has been deleted and only code that focuses on Abstract Data Type principles are remaining. The code is no longer able to compile due to the removed code. When follow the code it is seen that the Main program simply calls on the TriangleClass class. When the TriangleClass class is called it extends the GeometricObject class for its use of defining what a triangle is. This is an example of Abstract Data Types because the triangle is a type of geometric object. Therefore the GeometricObject class provides all the common options and settings for geometric objects, than the triangle object, TriangleClass, provides the specific information for the triangle being initiated in the Main program. This code displays the core principles of Abstract Data Types. As menitioned earlier, no exact code can show the principle, but this code shows the theory in practice. This code was created in NetBeans in the Java language. This program was originally three separate Java Files within the Source Package. The original file names are listed within the code comments in the Code Example below.

Code Example for Abstract Data Types

```
package bradshawlab;
//David Bradshaw Abstract Data Type Example
//Files cut to smaller size, no longer possible to compile

//Original File Called TriangleProject.java

public class TriangleProject {

   public static void main(String[] args) {

       TriangleClass Triangle = new TriangleClass(1, 1.5, 1);
   }
}
//Original File Called GeometricObject.java

public class GeometricObject {

  private String color = "white";
  private boolean filled;
  private java.util.Date dateCreated;
  }

//Original File called TriangleClass.java

public class TriangleClass extends GeometricObject {

    private double side1 = 1.0;
    private double side2 = 1.0;
    private double side3 = 1.0;

    public TriangleClass(double side1, double side2, double side3) {

        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

}
```

Declaration of Abstract Data Types in Programming

Every typed language required the programmer to declare the value being initialized. This is based on the principle of Abstract Data Types. By requiring the programming language to have a declaration for user defined values the program can be sure to always treat the values correctly and only allow operations that are compatible with the data type to be executed. Every language initiates values differently; however, they are all theoretically the same. This is possible because most languages use the same general syntax and keywords (Sebesta, 473-477). For example, most languages use some form of the word 'integer', whether it is 'int' or the full word spelled out for the declaration. Most other value types work in the same way, like 'character' can be 'char' or the full word. Types like 'String', 'Long', 'Boolean', and other specifications for the visibility outside the main program, like 'private' and 'public' are also usually written out in their full name.

When a programmer declares a value in the language of their choice, the program uses the keyword used, for example 'int', to find the code already created for the language; this could be listed in the API of the language in question. Once the language finds the declaration keyword in its repositories, the language will treat the variable as that value. If the programmer tries to put a non-compatible character in the declaration the compiler will throw an error, for example creating an integer and initiating it with the letter 'c'. Modern compilers will also catch errors and incompatible operations on certain declarations within the editor, so you can see the error while you type and not worry about compiling and getting an error. Below is an example of common declaration values that are examples of Abstract Data Types, these values call upon a saved procedure within the API or repository of the program language. The examples below are from the Java language and show their respective declaration keyword ("Types, Values, and Variables"). The values below are from Sun Microsystem's Java API and reference pages.

Chart of Data Declarations in Java

Type of Data	Declaration Keyword	Description
Integer	int	Most common declaration for numbers
Long	long	Inclusive declaration of longer numbers
Byte	byte	Small number, value from -128 to 127
Float	float	Decimal point numbers, floating decimals
Boolean	boolean	Logical System of True/False
Character	char	Letter or character, '\u0000' to '\uffff'
String	String	String of characters to create string
Object	object	Class instance or array in Object Oriented

Abstract Data Types effect on Object-Oriented Programming

There are many similarities between Abstract Data Types and Object-Oriented Programming. The main similarity is the idea that they both use abstraction for their core principles of programming. The code snippet above, “Code Example for Abstract Data Types”, shows the use of Object-Oriented Programming in Java. The code pulls data from a different class, or object, to another class that uses what is already known to fill in the knowledge of what is needed for the specific item. While Abstract Data Types and Object-Oriented Programming are alike there are a few differences. In Object-Oriented Programming the constructors are the main focus of what is being abstracted, while in Abstract Data Types the operations on the data are the significant item being handled (Cook).

Both Object-Oriented Programming and Abstract Data Types come from the same origin, SIMULA 67 (Cook). Both concepts are key parts of modern day programming and are used in every program created in today’s world. Object-Oriented Programming is a clear evolution of the principles of Abstract Data Types. As seen from the code snippet above, with the use of an Object-Oriented base program as an example of Abstract Data Types. Both Object-Oriented Programming and Abstract Data Types work in the general same manner. Object-Oriented Programming stores settings created by a user for a parent hierarchy and then allows the lower, or children, hierarchy levels to pull the information needed from the parents in order to execute there functions or operations. Abstract Data Types work in the same manner, but are used on a more fundamental way. When a programmer specifies that an integer value is being used the programming language calls out to the parent hierarchy for information on how to handle integers. When the child hierarchy receives the code for how an integer works it can see what operations are possible on the value, as well as the rules that can be applied to the value, like what values are possible for the value. Abstract Data Types are often referred to user-defined data types, since they are mostly defined by the user to allow them to alter and operate on values that are declared in their program. Without this declaration programs would not know how to handle different types of data and this would result in programs being very complex, as they would have to program the basic operations, like addition, subtraction and other numeral functional operations (Cook).

Conclusion

Summary of Abstract Data Types

In conclusion, Abstract Data Types are an amazing concept that makes life easier for programmers (Sebesta 470). It is seen that without Abstract Data Types programming would be a very complex task that require many lines of code to accomplish simple tasks. It is also seen how Abstract Data Types influence and evolution has allows for high level programming languages, like Java and Objective C, to allow for Object-Oriented Programming, which allows programmers to create extensions and hierarchy of classes that are required for multiple purposes, without the need of repetition for extended traits (Cook). Abstract Data Types are the core of modern programming, and are seen in every typed language being used. Abstract Data Types allow for easy programming and ‘behind the scene’ operations of type functionality. Abstract Data Types allow for programs to have declared values that can pull functional operations and ensure that the values pass the required level to be called the declared type.

My Opinion on Abstract Data Types

As seen, Abstract Data Types are used by every programmer and every program and are rarely given the appreciation they deserve. Before this paper, I have never heard of Abstract Data Types. However, after researching the topic and viewing examples, I quickly realized that I have been using the theory all along. Every time I declared a value in Java, I was actually implementing the principle of Abstract Data Types. I never questioned how Java knew to use addition and subtraction on an integer, but not on a character. I was also amazed to see the influence of Abstract Data Types on Object-Oriented Programming. I have programmed with Object-Oriented based classes in the past, but never once grasped the similarities of how my extension of hierarchy classes and my call for an integer value were alike.

What I learned

I feel that I have a deeper knowledge of what Abstract Data Types are and programming in general. After studying deeper into how declaration values are processed and how operations are effected based on their hierarchy, and being able to visually view what is being referred by studying the transportation of data amongst the Object-Oriented Programs that I have written in the past. I have learned quite a bit about how programming languages process data and how the inner workings of the programming languages work. I feel that the research papers and articles I found about Abstract Data Types were great key insights into the topic and showed how other professionals and scientist view and decipher the theory that is Abstract Data Types.

Works Cited

"Abstract Data Type." Encyclopædia Britannica. Encyclopædia Britannica Online. Encyclopædia Britannica, 2011. Web. http://www.britannica.com/EBchecked/topic/1086575/abstract-data-type.

Cook, William. "Object-Oriented Programming Versus Abstract Data Types." REX Workshop/School on the Foundations of Object-Oriented Languages (1990): 151-178. Web. http://www.cs.utexas.edu/~wcook/papers/OOPvsADT/CookOOPvsADT90.pdf.

Dale, Nell, and Henry Walker. Abstract Data Types: Specifications, Implementations, and Applications. 1st Edition. Lexington: D.C. Heath and Company, 1996. Google eBook.

Mitchell, John, and Gordon Plotkin. "Abstract Types have Existential Type." ACM Transactions on Programming Languages and Systems 10.3 (1988). Web. http://theory.stanford.edu/~jcm/papers/mitch-plotkin-88.pdf.

Sebesta, Robert. Concepts of Programming Languages. 8th Edition. New York: Pearson Education, 2008. Print.

"Types, Values, and Variables." Java Language Specification, Third Edition. Sun Microsystems, Inc., 2005. Web. http://java.sun.com/docs/books/jls/third_edition/html/typesValues.html.