

# I:Users:a540788:Downloads:java.png Notes

1. Next class November 9th 2015.

# I:Users:a540788:Downloads:java.png Creating and using structured data types

## Vector

A vector is very similar to an array with the difference that it can grow dynamically. Let us use a vector in Java:

// initialize and print the values of array

Vector<Integer> intVector = **new** Vector(10, 5);

**for** (**int** i = 0; i < intVector.capacity(); i++) {

intVector.add(2\*i);

System.***out***.print(intVector.elementAt(i) + "\n");

}

// enumerate through an iterator

Enumeration<Integer> enumerator = intVector.elements();

**while**(enumerator.hasMoreElements()){

System.***out***.print(enumerator.nextElement() + "\n");

}

The initial size of the vector is set at 10 and when the array grows dynamically the capacity (size) increases in increments of 5. We use an iterator pattern to cycle through the values of the vector. Note that one has to use the reference types (not primitive types) when using the generic version of vector, so Vector<Integer> will not give a compile-time error, but Vector<int> will give a compile-time error.

## Dictionary and Hashtable

A dictionary is a data structure used to store key-value pairs. The key provides an identifier to retrieve the value quickly (the fastest retrieval time, called random-access). The Dictionary class is an abstract class that cannot be instantiated. The implementation of the dictionary implemented in the example below is a hashtable, there are other types of dictionaries, such as a linkedhashtable data structure.

Dictionary<String, String> zipCodes = **new** Hashtable<String, String>();

zipCodes.put("Mark", "01877");

zipCodes.put("John", "01567");

zipCodes.put("Jen", "01444");

Enumeration<String> values = zipCodes.elements();

**while** (values.hasMoreElements()){

System.***out***.println(values.nextElement());

}

Enumeration<String> keys = zipCodes.keys();

**while**(keys.hasMoreElements()) {

String key = keys.nextElement();

System.***out***.println(zipCodes.get(key));

}

# I:Users:a540788:Downloads:java.png Java libraries and documentation

[www.docs.oracle.com/javase/8/docs/api](http://www.docs.oracle.com/javase/8/docs/api)

# I:Users:a540788:Downloads:java.png Classes, Objects and the meaning of static

Below is a typical definition of a class:

[public | private | protected] [abstract | final] class class-name

{

// data members (both primitive and objects) and member functions

}

A public class can be accessed by other classes in other packages, a private class can only be accessed by the class members, a protected class can only be accessed by other classes in the same package, or subclasses (derived classes). An abstract class cannot be instantiated, and is essentially an interface. A final class cannot be subclassed (derived) by other classes.

A class is composed of members and a member can be created following the syntax below:

[public | private | protected] [final | volatile] [static ] [transient]

Data-type variable-name;

The public member can be accessed through the object that instantiates the class, a private can only be accessed by other private members and methods, a protected can only be accessed by derived classes. A final member can only be set once, and it is a way to create constants. A volatile member has synchronized access by threads, which means that it is as if it was surrounded by a thread synchronization block. A static member is the same for all objects that instantiate the class. A transient member is a member which does not get serialized.

The syntax to create a member function is:

[public|private|protected] [abstract]|[final][static][native][synchronized]

  return\_type method\_name (arguments){

  // body}

If a method is public it can be accessed from the instance of the class, however if private it cannot be accessed by the object, but only by members of the class. Similarly a protected method cannot be accessed by the object, but can be accessed by members of the class and also subclasses. An abstract method belongs to an abstract class (they go hand in hand) and has no implementation, the implementation is reserved to the derived class. A final method is a method which cannot be reimplemented in the derived classes. A static method is a method that can be accessed by any instances of the class, and the body of a static method cannot access members of a class. A method is native when the implementation of the body is reserved to a C language library, and finally it is synchronized when access to the method is thread-safe.

# I:Users:a540788:Downloads:java.png Working with packages

A Java package is a technique for organizing classes into namespaces providing modular programming.Packages can be stored in compressed files called JAR files, allowing classes to be downloaded faster as groups rather than individually.

# I:Users:a540788:Downloads:java.png Encapsulation benefits

When creating a class, one hides the complexity of the implementation within the private and public methods and within the constructor of the class. One exposes members of the structure, or class through getters and setters. This hiding of complexity is referred to as encapsulation.