The need for Programming Languages

Computers are very simple devices in that they only understand a handful of simple commands, like adding two numbers or reading a value from memory. The set of commands understood by a computer is called *machine code*.

The processor is the component of a computer that executes the commands in a program. Each processor has its own machine code.

A program needs to be stored in the memory of the computer, so it can be executed. Information is stored in a computer in binary format, i.e, information is encoded as a sequence of 0’s and 1’s.

Below is a binary program in machine code for an old processor called 8086. This program prints the word “hello” on the screen.

```
10111010000011000000000011011010000001001110011
01001000011011100000000000100110011001101001000
00010100100001100101011011000110110001101111100
10110000100000010101111011011111011100100110110 0
01100100000100001000011010000110001101111
```

Binary code is hard for humans to understand, as a sequence of 0’s and 1’s has no meaning to us.

Compare the above binary program with the following equivalent python program

```
print ("hello")
```

Python is called a high level programming language and it was designed to make computer programs readable to humans. However, a computer does not understand python, java, C++, or any other high level programming language.

A compiler is a piece of software that translates programming language into code that the computer can understand.

In this course we will be writing programs in java. A java program must be stored in a file with the extension .java. A java compiler does not directly produce machine code, but it translates the java program into another language called *java bytecode*. Java bytecode is a kind of *intermediate language*.

Java bytecode is stored in files with the extension .class. A java *interpreter* or *virtual machine* can execute the java bytecode.

Eclipse has an integrated java compiler that runs as you type your program. If you want to compile your java program for a terminal or command window, the name of the java compiler is javac and the name of the java interpreter is java.

The following two classes are used to illustrate the notions of instance variables and public and private methods. Class Person.java represents objects, each one of which stores the first name, last name, and email address of an individual. Setter and getter methods are used to access the private information stored in an object of this class.

Class SocialNetwork.java has an array instance variable called listFriends which can store a set of objects of the class Person. Method *add* allows a new Person object to added to the array. Notice that the size of the array is increased as more data items

are stored in it. Method expandCapacity() doubles the size of the array and copies the information from the smaller array to the larger one.

```
public class Person {
        /* Attribute declarations */
        private String lastName;              // last name
        private String firstName;             // first name
        private String email;                 // email address

        public Person() {   /* Constructor */
                lastName = “”;
                firstName = “”;
                email = “”;
        }

         /* Constructor initializes the person's name and
           email address */

         public Person(String first, String last, String mail) {
                 firstName = first;
                 lastName = last;
                 email = mail;
         }
```

```
  public String getName() {
            return firstName + " " + lastName;
}
  public String getEmail() {
            return email;
  }
  public void setEmail (String mail) {
            email = mail;
  }
  public void setName(String newFirst, String newLast) {
            firstName = newFirst;
            lastName = newLast;
  }

   public String toString() {
             String s = firstName + " " + lastName + "\t" + email;
             return s;
    }
```

```
    /**
     * equals method determines whether two Person objects
    * have the same name
     * @param other:     Person object that this is compared to
     * @return true of they have the same first name and last
     *  name, false otherwise
     */
    public boolean equals(Person other){
            if (this.firstName.equals(other.firstName) &&
                        this.lastName.equals(other.lastName))
                    return true;
            else
                    return false;
    }
}
```

```
public class SocialNetwork {

        // default size of array
        private final int DEFAULT_MAX_FRIENDS = 10;

        /* Attribute declarations */
        private Person[] friendList; // list of friends
        private int numFriends;                 // current number of
                                                // persons in list


        /**
         * Constructor creates person array of default size
         */
        public SocialNetwork () {
                friendList = new Person[DEFAULT_MAX_FRIENDS];
                numFriends = 0;
        }
```

```
/**
 * Constructor creates person array of specified size
 * @param max maximum size of array
 */
public SocialNetwork(int max) {
        friendList = new Person[max];
        numFriends = 0;
}

public void add (String firstName, String lastName, String email) {
        Person friend = new Person(firstName, lastName, email);

        // if array is full, increase its capacity
        if (numFriends == friendList.length)
                expandCapacity();

        // add new friend at first free entry in array
        friendList[numFriends] = friend;
        numFriends++;
}
```

```
/**
* expandCapacity method is a helper method
* that creates a new array to store friends with twice the
* Capacity of the existing one
*/
private void expandCapacity() {
        Person[] largerList = new Person[friendList.length * 2];
        for (int i = 0; i < friendList.length; i++)
                largerList[i] = friendList[i];

        friendList = largerList;
}

public String toString() {
        String s = "";
        for (int i = 0; i < numFriends; i++) {
                s = s + friendList[i].toString()+ "\n";
        }
        return s;
}
```

Class SocialNetwork contains a method for removing a data item from the array. To remove a data item, say *target*, from the array we first need to find the position of such an item in the array. A simple way of looking for *target* in array *friendList* is to take the data items stored in the array one by one starting at the data item stored in index 0 and compare each one of them with *target* until either

- *target* is found, or
- all data items have been examined and *target* is not found

The above algorithm for looking for a data item in a list is called *linear search*.

Once item *target* has been found in the array we can remove it by replacing it with the last item in the array. Pseudocode for removing a data item from the array follows.

Algorithm remove(*target*)
Input: data item to be removed
Output: true if *target* was removed from the array; false
if *target* was not found in the array

```
i = 0
while (i < numFriends) and (friendList[i] not equal target) do
        i = i+1
if i = numFriends then return false
else {
        friendList[i] = friendList[numFriends-1]
        friendList[numFriends-1] = null
        numFriends = numFriends -1
        return true
}
```

The advantage of writing an algorithm in pseudocode is that we can concentrate on designing the steps that the algorithm needs to perform to achieve the desired task without having to think about how to express the algorithm in correct java syntax.

Once we have designed a correct algorithm for a problem in pseudocode, translating it into java is a somewhat mechanical process.

Writing algorithms in pseudocode and then translating them into java makes it easier to design programs.

The beauty of pseudocode is that there is no fixed syntax or rigid rules for it. Pseudocode is a mixture of English and programming-like statements.

Each programmer comes up with their own version of pseudocode. The programmer just needs to ensure that pseudocode is understandable to other people and that it is detailed enough that translation into java or other programming language is simple. There should be an (almost) one-to-one correspondence between lines of pseudocode and lines of java.

The java version for the remove algorithm follows.

```
public boolean remove(String firstName, String lastName) {
        Person target = new Person(firstName, lastName, "");

        // search the list for the specified friend
        int i  = 0;
        while ((i < numFriends) && !friendList[i].equals(target))
                i++;

        if (i == numFriends) return false;
        else {
                // person found, remove by replacing with last one
                friendList[i] = friendList[numFriends - 1];
                friendList[numFriends - 1] = null;
                numFriends --;
                return true;
}
```

```
public class MyFriends {
  public static void main(String[] args) {

    SocialNetwork contacts = new SocialNetwork();

    contacts.add("Snoopy","Dog","snoopy@uwo.ca");
    contacts.add("Felix","Cat","felix@uwo.ca");
    contacts.add("Mickey","Mouse","mickey@uwo.ca");

    System.out.println(contacts.toString());
  }
}
```

The second line in the above class:

```
public static void main (String[] args) {
```

states that method main, when invoked can receive any number of arguments. Java will create an array of the correct size to store all the arguments. Java was designed so that the first method that is executed in any java program is main. The arguments that are passed to this method are called the *program arguments* or *command line arguments*.

The keyword static tells the java compiler that when the program is executed a special object, sometimes called a *static object,* needs to be created. Static objects are different from other objects in that they are not created by the programmer with the new statement. Since to run a java program objects need to be created first, static objects solve the issue of how to create the very first object of a program.