

Welcome to the AITP NCC 2000 Java Competition in Tampa, Florida.

Context:

Big Tex's Taste of Texas is a purveyor of fine Texican food throughout the United States. Recently, he has been getting orders from Sweden for his authentic, home style Texican food. Tex's current customer address system is set up for US addresses and he needs the capability to add Swedish addresses.

Your team, along with two others, has been assigned to add Swedish address functionality as follows:

- Your team is assigned to implement the **Application** and **SwedishAddress** classes (Java Competition Team).
- A second team has just completed modifications to the GUI, which your classes will work with (Scenario - No Actual team).
- A third team has just completed the modification to the **Address** abstract class, which your classes will also work with (Scenario - No Actual team).

This is a parallel development project. An object-oriented architectural design team has provided a UML model and required documentation for you to follow. You will find the resulting model and generated code framework attached to this document.

Main Requirement Specification

- You are required to implement the **IAddress** interface. Failing to do so will disqualify your team.
- Your delivery must support all existing functionality that has been built into the product.

You must provide the following services in the **SwedishAddress** class:

- Add US addresses
- Change-edit-update US addresses
- Add Swedish addresses
- Change-edit-update Swedish addresses
- Change an existing US address into a Swedish address
- Change an existing Swedish address into a US address
- Delete an address

You will find project support files at the following mapped network drive locations:

| | |
|-----------------|-------------------------|
| AITPjava | Problem statement files |
| AITPjavaSupport | Resource files |

To work with these files, you will need to download the appropriate files to your local hard drive. You will deliver your product to the customer on a 3.5-inch diskette.

For additional details, refer to the online documentation that will be introduced during the project briefing.

The Java Contest consists of two parts:

Part 1: Using the UML diagram provided, edit **Application.java** to start up the application properly. You will have 15 minutes to solve this part.

Part 2: Implement the **SwedishAddress.java** solution described above using the UML diagram provided.

Good Luck!

```

/**
 * Interface Name      : IAddress.java
 * Version             : Build 01.00
 * Date of Change     : 2000 MAR 07
 * Author              : Team #
 * Copyright           : 2000 ASR Strategic Resources
 *                    : AITP NCC 2000 Java Competition
 * Description         : An interface that sets the ground rules for the
 *                    : rest of the address classes to follow.
 * Version History :
 *
 * Version      Date      Author
 * Build 01.00  2000 MAR 31  Team #
 *                    Added functionality.
 * Build 00.00  2000 MAR 31  Don Baldwin
 *                    Provided Interface signature.
 *                    Do not modify this class - doing so is a
 *                    disqualifying event!
 *                    Note: Each team must place their team number
 *                    in the two locations indicated above.
 *
 * Known Problems:None
 */

import java.sql.SQLException;

public interface IAddress
{
    // Methods for set and get the Id
    public void setId(long aLongID) throws IllegalArgumentException;
    public long getId();

    // Methods for set and get the FirstName
    public void setFirstName(String aStrFirstName) throws
        IllegalArgumentException;
    public String getFirstName();

    // Methods for set and get the LastName
    public void setLastName(String aStrLastName) throws IllegalArgumentException;
    public String getLastName();

    // Methods for get the FirstLastName and LastFirstName which means,
    // Example: Don Baldwin and Baldwin, Don
    public String getFirstLastName();
    public String getLastFirstName();

    // Methods for set and get the CompanyName
    public void setCompanyName(String aStrCompanyName);
    public String getCompanyName();

    // Methods to get the informations about the address
    public String getStreet();
    public String getCountry();
    public String getCountryCode();
    public String getMailingLabel();
}

```

```
// Postal code methods.
public void setPostalCode(String aStrPostalCode) throws
    IllegalArgumentException;
public String getPostalCode();
public void setPostalRegionCode(String aStrPostalRegionCode) throws
    IllegalArgumentException;
public String getPostalRegionCode();
public void setPostalBlockCode(String aStrPostalBlockCode) throws
    IllegalArgumentException;
public String getPostalBlockCode();

// All addresses must know how to add themselves
// to the database.
public void writeAddress() throws SQLException;

// A general toString method.
public String toString();
} // END IAddress.java
```

```

/**
 * Class Name       : Application.java
 * Version          : Build 01.00
 * Date of Change   : 2000 MAR 31
 * Author           : Team #
 * Copyright        : 2000 ASR Strategic Resources
 *                 : AITP NCC 2000 Java Competition
 * Description      : This is the main class for the AITP NCC Address
 *                 : project. It starts the MainFrame.class component.
 *
 *                 MAIN CLASS: AITP NCC Address Project
 *
 *
 * Version History:
 *
 * Version      Date      Author
 * Build 01.00  2000 MAR 31  Team #
 *
 *                 Added functionality.
 * Build 00.00  2000 MAR 31  Don Baldwin
 *
 *                 Removed functionality from class.
 *                 Contestants will need to add either three
 *                 lines or 1 line depending on how they
 *                 choose to call the MainFrame.class.
 *                 Hint... Think 3! It's the OO way!
 *
 * Known Problems: None
 *
 *                 Especially after all of the functionality was removed,
 *                 unless you consider that a problem... :)
 */

// NOTE: Add appropriate comments to source code, and remove this line!
public class Application
{
    public static void main(String[] args)
    {
        // Remove the following line (And remove this comment line)!
        System.out.println("Please edit Application.java to make this run");

        // Insert your code here (And remove this comment line)!
    }
} // END Application.java

```

```

/**
 * Class Name       : SwedishAddress.java
 * Version          : Build 01.00
 * Date of Change   : 2000 MAR 11
 * Author           : Team #
 * Copyright        : 2000 ASR Strategic Resources
 *                  : AITP NCC 2000 Java Competition
 * Description      : The address objects
 *                  : specific for Swedish
 *                  : addresses.
 *
 * Version History:
 *
 * Version          Date          Author
 * Build 01.00      2000 MAR 31   Team #
 *                  Added functionality.
 * Build 00.00      2000 MAR 31   Don Baldwin
 *                  Provided design framework construct.
 *
 *                  Add functionality to the framework skeleton
 *                  that is provided below. You will need to
 *                  make appropriate modifications to the code
 *                  below.
 *
 * Known Problems: None
 */

```

```

import java.sql.SQLException;

public class SwedishAddress extends Address
{
    /**
     * Method to set the street information.
     */
    public void setStreet(String aStrStreet)
        throws IllegalArgumentException
    {
    }

    /**
     * Method to get the street information.
     */
    public String getStreet()
    {
        return "Not implemented yet";
    }

    /**
     * Method to set the country code.
     */
    public void setCountryCode(String aStrSECountryCode)
        throws IllegalArgumentException
    {
    }

    /**
     * Method to set the country code.
     */
    public String getCountryCode()
    {
        return "Not implemented yet";
    }
}

```

```

    }

    /**
     * Method to set the country name.
     */
    public void setCountry(String aStrSECountry)
        throws IllegalArgumentException
    {
    }

    /**
     * Method to get the country name.
     */
    public String getCountry()
    {
        return "Not implemented yet";
    }

    /**
     * Method to set the city name.
     */
    public void setCity(String aStrCity)
        throws IllegalArgumentException
    {
    }

    /**
     * Method to get the city name.
     */
    public String getCity()
    {
        return "Not implemented yet";
    }

    /**
     * Method to set the postal region code.
     */
    public void setPostalRegionCode(String aStrPostalRegionCode)
        throws IllegalArgumentException
    {
    }

    /**
     * Method to get the postal region code.
     */
    public String getPostalRegionCode()
    {
        return "Not implemented yet";
    }

    /**
     * Method to set the postal block code.
     */
    public void setPostalBlockCode(String aStrPostalBlockCode)
        throws IllegalArgumentException
    {
    }

    /**
     * Method to get the postal block code.
     */
    public String getPostalBlockCode()
    {

```

```

        return "Not implemented yet";
    }

/**
 * Method to set the postal code.
 */
public void setPostalCode(String aStrPostalCode)
    throws IllegalArgumentException
    {
    }

/**
 * Return the postal code as "RegionCode BlockCode"
 * (xxx xx)
 */
public String getPostalCode()
    {
    return "Not implemented yet";
    }

/**
 * Build the mailing label which will be displayed like
 * this in the address book.
 */
public String getMailingLabel()
    {
    return "Not implemented yet";
    }

/**
 * Creates a new european address in the database with
 * the data retrieved from the address object passed to
 * the method.
 */
public void writeAddress()
    throws SQLException
    {
    }

/**
 * toString() displays the class attributes,
 * for debugging purposes.
 */
public String toString()
    {
    return "Not implemented yet";
    }

} // END SwedishAddress.java

```

After Contest Notes

This Java Contest problem statement is designed to introduce the contestants to a real-world situation. Industry best-practices dictate that prior to coding a solution, an analysis effort be undertaken to produce a design. The resulting design is delivered to the programmers in the form of a UML model. This UML model can be transformed into a Java program stub framework in which the method stubs are provided to the programmer for implementation.

The scenario given in the contest problem statement, in which three teams were described as being used to build the solution, was actually taken by the programmers that developed this problem statement. Parallel development is common in industry, which is why UML skills are an essential part of the software development process. By using a stub framework, parallel development can take place without creating compiler errors. As functionality is implemented, the stubbed framework becomes a functioning class.

This problem statement was reviewed by both faculty and industry practitioners to make sure that it could be reasonably solved by students that have completed a first semester course in Java. Anyone studying Java should have attended a course in Object-Oriented software design based on the Unified Modeling Language (UML). Prerequisites for attending a Java programming course should have included a structured programming course, an object-oriented design course, and a database course. A computer network course would also be helpful. Java is a design-centric programming language and should not be treated as a stand-alone introductory course since there are too many critical aspects that need to be understood. The amount of conceptual information required by Java makes it unsuitable as a stand-alone introductory course to programming.

Java Contest Coordinator Notes

The team that was awarded first place for their solution to this problem statement turned in a fully functional solution. The second and third place teams also turned in fairly complete solutions. The honorable mention team was on the right track and was meeting many of the requirements. The problem statement was designed so that approximately 25% of the participating teams should be able to deliver a solution. This was the first ever AITP NCC Java competition and the expected percentage of teams did provide solid solutions to the problem statement.

The overall problem statement is designed as a case study that can be used in Java programming instruction. The design team took several programmatic approaches so that the instructor can compare different programming techniques. In interviewing a number of Universities and technical schools, the problem statement design team feels that this problem statement represents an achievable level of Java programming competence for a first semester Java class. The problem statement also represents a minimal level of Java competence expected from industry.

Please address questions and comments to:

don.baldwin@pobox.com