Carnegie Mellon University
Silicon Valley

Programming Diagnostic

Purpose

The purpose of the diagnostic is to provide a context for prospective students to demonstrate proficiency in a programming language and basic object-oriented concepts. While programming is not the main focus of the degree programs, there is some programming involved.

During your interview, you will be asked programming questions in one of the following languages. You will inform your interviewer during the interview which language you are strongest in.

- C
- C++
- C#
- Java
- Python
- Php
- Ruby on Rails

Before your interview is scheduled, you will need to complete the following programming diagnostic. This diagnostic can be done in any object oriented programming language using any frameworks with which you are familiar.

Directions

1. Build an application that implements the requirements described below. You can use any tools or libraries or plug-ins that you see fit.
2. You may use any object oriented language that supports either a desktop user interface or a web-based framework.
3. Here is a list of frameworks that we routinely see used for this diagnostic, listed in alphabetical order:
    - Android
    - Google App Engine
    - Java JSP, servlets, struts, Jboss Seam
    - Java Swing
    - Mac OS X Cocoa or iPhone's iOS
    - Ruby on Rails
4. If you decide to use Swing, you are encouraged to use a GUI builder

5. When you are finished with the diagnostic, email a zip file containing the completed code and directions for installing and running it to diagnostic@sv.cmu.edu. Your zip file should include:
    - Name of the programming language used
    - the source code
    - any build files used to package your code (ie makefile, ant file, rakefile)
    - any libraries that you used (if the library is very large, then include the version of the library you used and where we can find it on the internet)
    - If you built a web application, there is no need to include the application server code.
    - If you did a Java swing implementation, please include an executable jar file. Please include directions for running the program without using an IDE.
6. **Important Note:** Our email system checks the contents of the file. **If you have executables**, **please rename the file to .doc before zipping it**. Please make sure that you include instructions on which file to rename back to what extension. For example: "Included in the zip file is work.doc. After downloading and unzipping the files, please rename work.doc to work.exe in order for this to work".
7. Contact Stacy Marshall at stacy.marshall@sv.cmu.edu or (650) 335-2852 to schedule your admissions interview.
8. If you will be on-campus and have a laptop, bring your laptop to the interview. If you don't have a laptop, we'll expect you to find a way to demonstrate your code to us.
9. If you will not be on-campus for the interview, you will need access to a computer and internet access during the interview.
10. Be prepared to demonstrate and discuss your code during the interview.
11. Be prepared to describe the class structure during the interview (You may want to sketch a class diagram ahead of time.)

Application Description

Build an application that allows a user to specify a travel request as shown in the figures below. Your implementation must use the Model-View-Controller (MVC) design pattern. Specifically, the data for the model should be contained in separate classes that are not part of the UI code. You should have at least two model classes - one class that defines the values for the combo boxes and another class that stores the data entered into the form by the user.

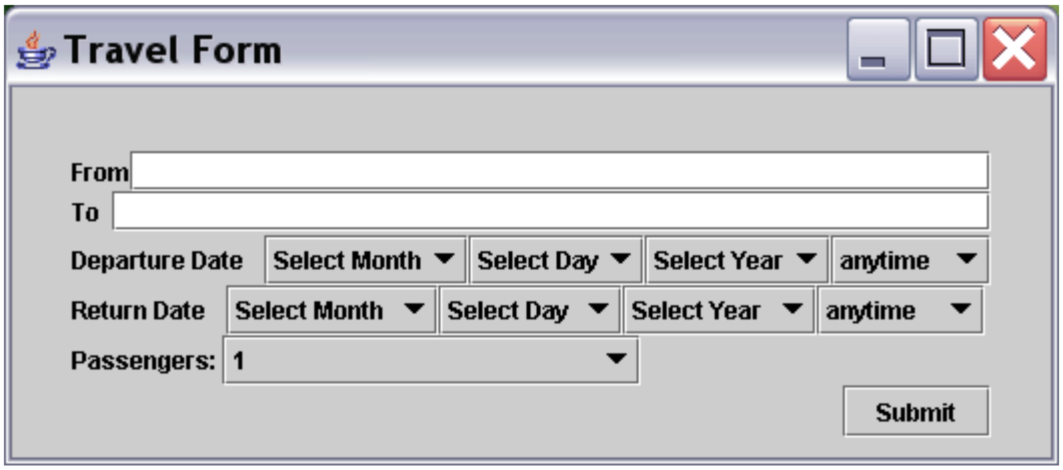
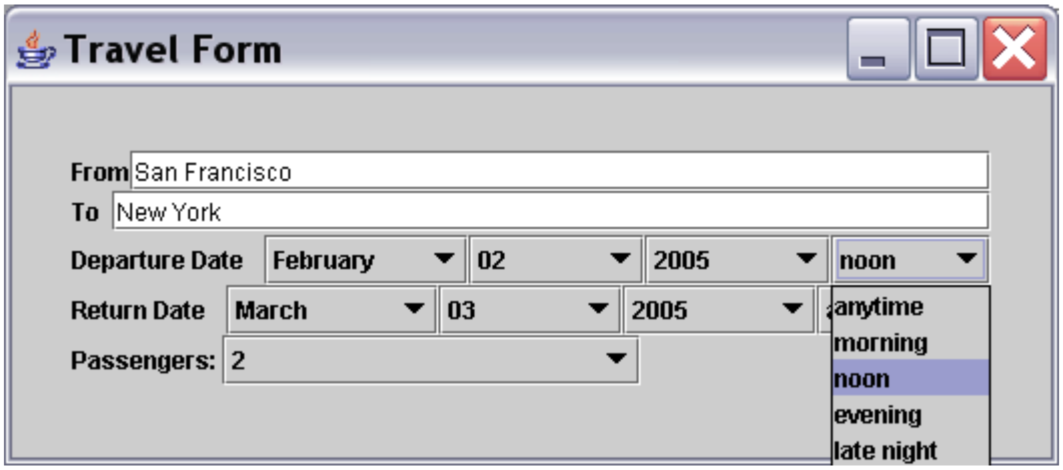
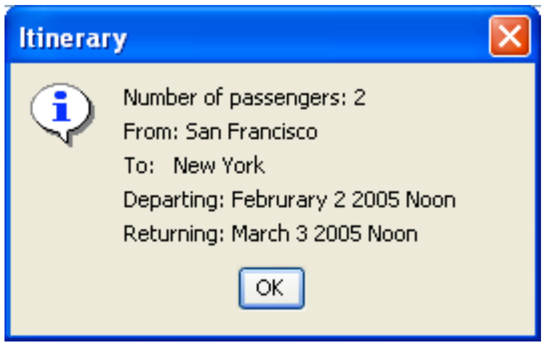


Figure 1: Screen shot of application when started



From San Francisco
To New York
Departure Date February 02 2005 noon
Return Date March 03 2005
Passengers: 2

Figure 2: Screen shot of application with data entered



Number of passengers: 2
From: San Francisco
To: New York
Departing: Februrary 2 2005 Noon
Returning: March 3 2005 Noon

Figure 3: Showing submitted data

Note:

The screen shots above are based on Java Swing technology, but you can use any OO programming language to implement this functionality.

Business rules

- The range of data for the year should be 2011-2012. The range of data for the number of passengers should be 1-10.
- When the user clicks the submit button, if all of the data fields are valid, the application should show the submitted data in a textual form (see Figure 3).
- When the submit button is clicked, if any of the fields are in an invalid state, an error should be displayed. An invalid state for each field is defined as follows:
    - From: blank data
    - To: blank data
    - Departure Date: The combination of the month/date/year fields should produce a valid date
    - Return Date: The combination of the month/date/year fields should produce a valid date
- The return date should be on or after the departure date. (We are not worried about crossing the International Date Line.)

Hint: your programming language probably already has a date class that is better tested than any date code you write for this assignment