

1. 1. (30) You have a soda machine that dispenses several different kinds of soda in exchange for money. You want to model a single transaction (a person buying a soda).

(10) Provide an analysis (numbers indicate how many are expected):

I. Inputs (2) :

II. Outputs :

A. Return values (2) :

B. Side effects (1) :

III. Constraints (2) :

IV. Assumptions (2) :

(10) Provide a flowchart that models a transaction – don't forget error conditions.

(10) Model the soda machine using UML. The soda machine should contain 2 types of sodas and 2 types of juices (of your choosing); note that “Soda” and “Juice” are abstract concepts and only their subclasses can be instantiated. All drinks in the machine should conform to the interface “Can” and implement the method “stack”. The machine should also contain Coins (specifically, Nickels, Dimes and Quarters). The machine itself should have a private variable totalInserted, a public method insertCoin which takes a parameter of type Coin, and a public method dispense with a return type of boolean. No other methods or member variables are required. Please indicate multiplicities on composite relationships.

2. (15) Suppose you have defined the exception below:

```
class InvalidInputException extends Exception {  
    public InvalidInputException(String message) {  
        super(message);  
    }  
}
```

Write a method “checkUserInput” which takes an argument of type “int” and throws an exception if the input is negative but otherwise does nothing.

Write another method “processUserInput” which takes an argument of type “int”, has a return type of “boolean”, and calls “checkUserInput”. This method should catch the exception thrown by checkUserInput, return true if no exception occurs and print out an error message and return false if an exception does occur. Print out the message “Input Phase Completed” (after checkUserInput is called) regardless of whether an exception occurs or not.

3. (15) Imagine that you have a file with 100 entries in the following format:  
<Last Name>:<First Name>:<age>:<social security number>

An example line would be as follows:  
Saunders: Billy:27:123456789

You are to read in this file and store the entries in an array of type Person.  
The definition for class Person is as follows:

```
class Person {
    private String lastName, firstName;
    private int age;
    private long ssid;

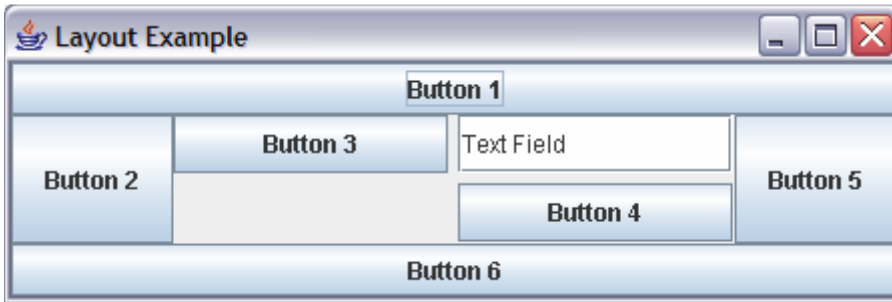
    public Person(String lastName, String firstName, int age, long ssid) {
        ...
    }
}
```

Complete the following method. You may use any I/O classes you see fit. Remember to check for exceptions.

```
public Person[ ] getPeopleFromFile(String fileName) {
```

```
}
```

4. (15) You are creating a subclass of JPanel that shows the widgets displayed below. You must select one or more layouts and use them to display the widgets in the correct order. Do not use absolute layout.



All the setup code has been completed for you, all that you must do is complete the following method:

```
// Sets the layout for this panel and adds the widgets using the layout.  
public void initWidgets() {
```

```
}
```

5. (30) You want to create a `CircularButton` widget. When the button is clicked with the mouse, you want it to fire an `ActionEvent` (using the `AbstractButton` method “`fireActionPerformed(ActionEvent e)`” – pass a null argument to this method) The button should be red, but should turn blue when depressed (mouse button has been pressed but not released). For this, you will add to the following class:

```
public class CircularButton extends AbstractButton {
    (3)

    public CircularButton(String text) {
        super();
        setText(text);
        initWidget();
    }

    public void initWidget() {
        (1)
    }

    public void paint(Graphics g) {
        (2)
    }
}
```



height = width  
 Label starting coords:  
 (width / 2, height / 2)  
 Text is black

You can assume `AbstractButton` has all the same functionality as a `JPanel`. You should use the `getHeight()`, `getWidth()`, and `getText()` methods. At (1), you should add the appropriate listener – use the distance formula to check whether the click/press is within the area of the circle. At (2), you should add the code to draw the button – be sure to check a flag to see which color the button should be. At (3), you should declare the flag. The flag should be set/unset in (1).

$$\text{distance} = \sqrt{((x - x_{\text{center}})^2 * (y - y_{\text{center}})^2)}$$

Write each of your code snippets (labeled with the numbers 1-3) on a separate page.