A picture containing text, clipart

Description automatically generated



**C# PROGRAMMING**

(330)

**REGIONAL 2024**

**PRODUCTION:**

Regional\_CSharp \_\_\_\_\_\_\_\_\_\_\_(510 points)

**Test Time: 90 minutes**

**GENERAL GUIDELINES:**

*Failure to adhere to any of the following rules will result in disqualification:*

1. Member must hand in this test booklet and all printouts if any. Failure to do so will result in disqualification.
2. No equipment, supplies, or materials other than those specified for this event are allowed in the testing area. No previous BPA tests and/or sample tests (handwritten, photocopied, or keyed) are allowed in the testing area.
3. Electronic devices will be monitored according to ACT standards.

You will have ninety (90) minutes to complete your work.

Your name and/or school name should *not* appear on work you submit for grading.

1. Create a folder on the flash drive provided using your contestant number as the name of the folder.
2. Copy your entire solution/project into this folder. The project folder for you has already been provided: Regional\_CSharp
3. Submit your entire solution/project so that the graders may open your project to review the source code.
4. Ensure that the files required to run your program are present and will execute on the flash drive provided.
5. You will need to use Visual Studios 2019 or greater to complete this exam.

\*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

The graders will *not* alter your source code. Submissions that do *not* contain source code will *not* be graded.

**Assumptions to make when taking this assessment:**

* The goal of this assessment is to create a Windows Form App that will allow the user manually to create a list of minerals to be mined.
* A Visual Studio project folder with the main project file is provided in the contest folder (ChamaBasinMiner.sln).
* There are interactive objects on the **Form1**. The form only contains a Text Box called **txtEntry** and a button called **btnEnterData**.
* There are no labels on the form, however you will be creating all of the labels through your code. In order to create the label, you will need construct it (**Label** class); set its XY coordinates, give it flexible sizing, and pass it a String object for its text.
* Objects can be added to the form with the code **Controls.Add(**Label Object**).**
* The object classes and Form1 classes have been created for you; however, you will still be required to program certain actions in those classes. In addition, the form design has been already created; however, you will add objects to it through your code.
* Your program will produce one calculation:
  + **Estimated Market Value:** Calculated in **Minerals.cs** for each mineral created in the form:
    - **Market Rate \* Total Tons.**

**Development Standards:**

* Your Code must use a consistent variable naming convention.
* All subroutines (if any), functions (if any), and methods (if any) must be documented with comments explaining the purpose of the method, the input parameters (if any), and the output (if any). Readability is a goal of good code.

**Commenting for Source Code Review (see the rubric):**

* Certain sections of your code will be graded. These gradable blocks of code can range from creating data structures, method algorithms, exception handling, and class construction.
* The grading rubric contains a section called Source Code Review: in this section are listed a description all the graded programming concepts.
* Each gradable item must have a comment listed at its beginning, and the comment must be prefixed with the comment flag. The flag helps the graders easily locate the code to increase the effectiveness of grading.
* The flag will always use this naming convention: **SC#** (NOTE: the # symbol will be replaced with sequential numbering, i.e. **SC1, SC2, SC3**, etc.
* No explanation in the comment with the flag is required, only the comment flag; however, any information placed in the comment could help the grader better understand and avoid any costly errors.
* The comment flag needs to be place in close proximity to the block of code it represents.
* If a comment flag is not present, you will not receive credit.
* In this example the Source Code Review has a gradable section of code for printing to the console (Remember these are non-related examples):
  + SC12: ***WriteLine*** method in the ***Form1*** classis printing the correct object \_\_\_\_ 10 pts
  + The user will place the code above the method call:

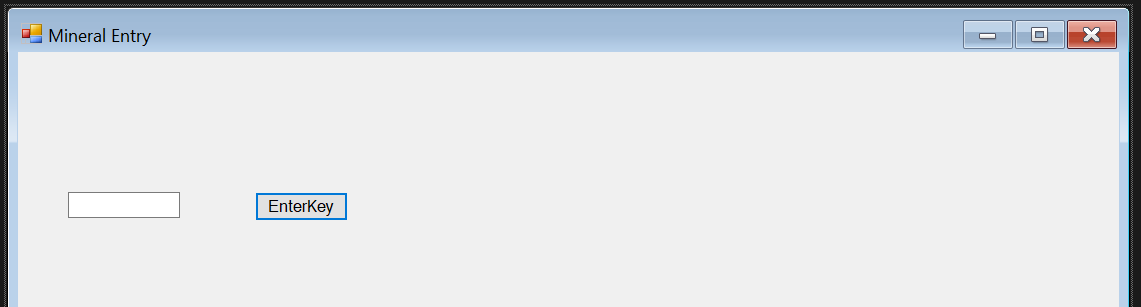
//SC12 printing a sarcastic comment

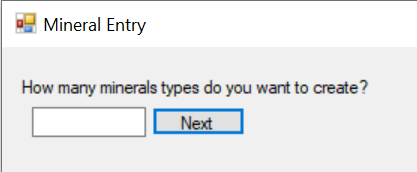
Console.WriteLine("Get ready for some fun!");

**Regional\_CSharp**

A mining company in New Mexico specializes in extracting minerals that are used to create lithium batteries. The company is wanting you to develop a prototype software to allow a user to manually create minerals for data entry. The form will collect how many they want to make, the mineral symbol, mineral name and how many tons were mined.

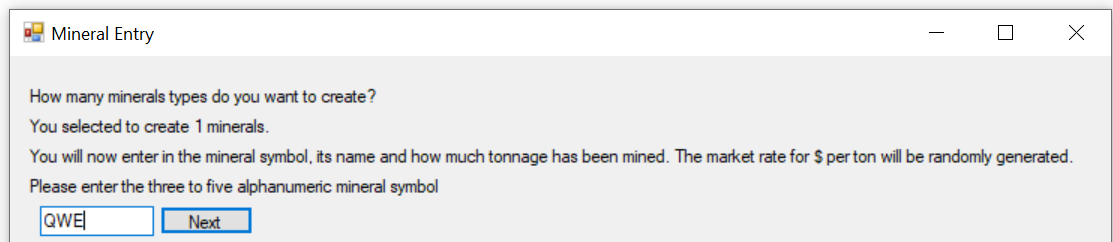
**Premade Form**

A premade form (see image below) has been created for you. It does not need to be changed. The form is called Form1 and it has two objects embedded on it: text box and a button. The text box and the button are dynamic and will change their location based upon which step the user is when entering data into the form. These objects use XY coordinates for their placement; you will only change their Y axis values with the same value (for example, you will increase both of the Y axis with increments of 20). NOTE: the XY coordinates of these objects in the design view will be changed through the code once the program starts.

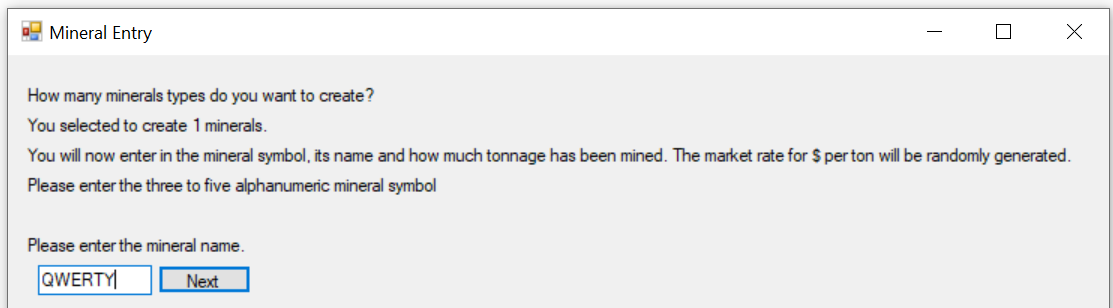
**Input/Output**

STEP 1: The input and out of the program are intertwined as the program will dynamically adjust based upon the user input. The first step will be the user entering how many minerals they want to create (image to the right). The entry must be a whole number.

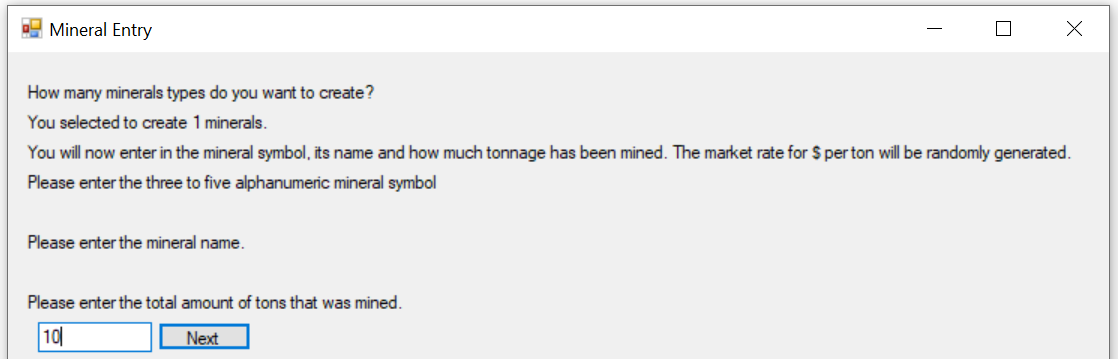
STEP 2: After a successful entry of how many minerals to create, the program will ask the user to enter in a 3 to 5 symbol set. The user can enter in letters, numbers or symbols.

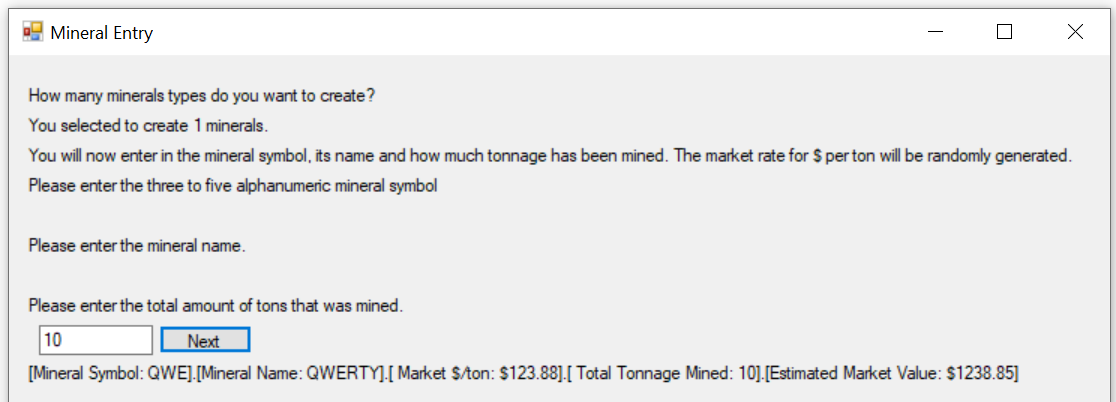


STEP 3: The user will enter in the mineral name. The user can enter in letters, numbers or symbols.



STEP 4: The user will enter the total amount mined. The value must be a whole number.

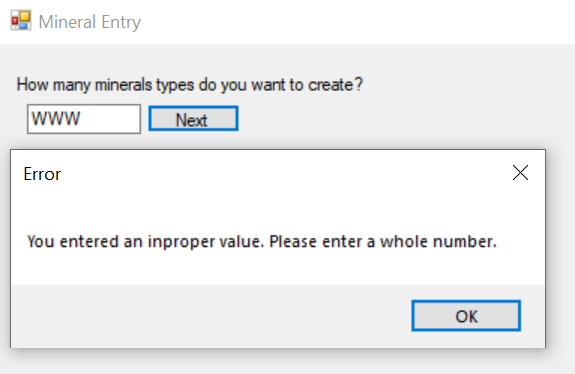


STEP 5: The final output will be printed on the form below the last user data entry. The output will be in a code generated label.

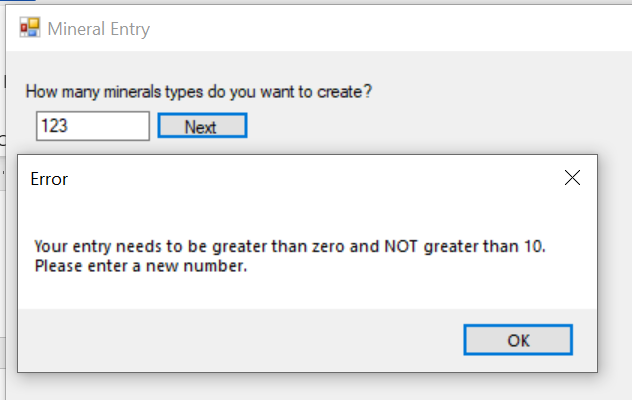
STEP 5 NOTE: if there is more than one mineral entered, the code will get all of the information for each mineral (STEPS 2-4), and then will print all of the entered information in a multiline label. See image below for a sample printout.



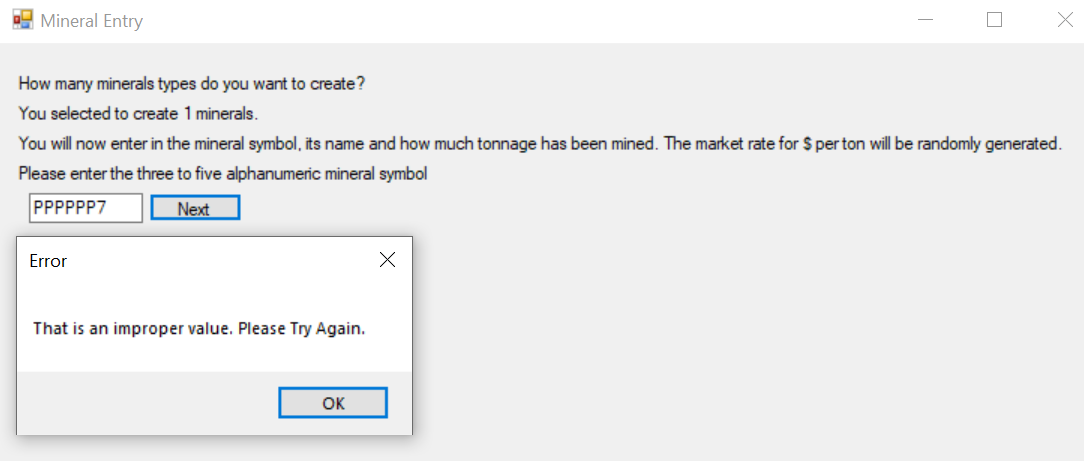
**Input Errors w/ Output Message Boxes (**NOTE: all error messages are Message Boxes)

STEP 1 Wrong Data Type: if the user enters in a non-whole number into the entry a message box will appear warning them of the entry error (see image to the right).

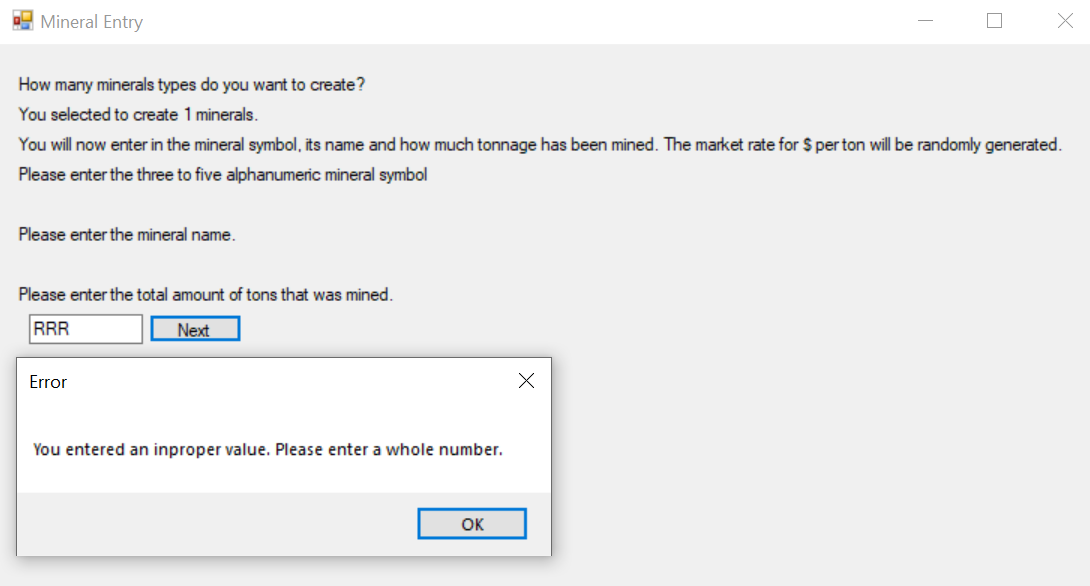
STEP 1 Range Violation: if the user enters in a whole number below or greater than the required range a message box will appear (see image below). The range is 0 to 10 inclusive.



STEP 3 Range Violation: If the user enters a symbol that is less than 3 alpha-numeric values (or symbols) or greater than 5, a message box will appear warning them of their error (see image below).



STEP 4 Wrong Data Type: if the user enters in a non-whole number into the entry, a message box will appear warning them of the entry error (see image below).



**Object XY Coordinates**

These are tips for the starting XY coordinates and their incremental adjustments. All the button and text entry objects will be aligned with the same X-axis coordinate of 20. The labels will all have an X position of 10. The Y-axis will vary, but all object Y positions will be a multiple of 20.

**btnEnterData**

* STEP 1 Program Start: (X:20),(Y:40)
* STEP 2 Second Entry: (X:20),(Y:100)
* STEP 3 Third Entry: (X:20),(Y:140)
* STEP 4 Fourth Entry: (X:20),(Y:180)

**txtEntry**

* STEP 1 Program Start: (X:100),(Y:40)
* STEP 2 Second Entry: (X:100),(Y:100)
* STEP 3 Third Entry: (X:100),(Y:140)
* STEP 4 Fourth Entry: (X:100),(Y:180)

**Labels**

* First Label STEP 1: "How many minerals types do you want to create?”: (X:10),(Y:20)
* Second Label STEP 2: "You selected to create " + value + " minerals.": (X:10),(Y:40)
* Third Label STEP 2: "You will now enter in the mineral symbol…”: (X:10),(Y:60)
* Fourth Label STEP 2: "Please enter the three to five alphanumeric mineral…”: (X:10),(Y:80)
* Fifth Label STEP 3: "Please enter the mineral name.": (X:10),(Y:120)
* Sixth Label STEP 4: “Please enter the total amount of tons…”: (X:10),(Y:160)
* Seventh Label STEP 5: This label will have the printout for all of the minerals created. Each mineral will appear on a separate line. (X:10),(Y:200)

**Requirements/Explanations**

1. You must use the provided Visual Studio project called “**ChamaBasinMiner**” in the Regional\_CSharp folder. This project will contain a “**Form1.cs[design]**” that has been already created in the organization and naming conventions.
2. No additional objects can be added to the form design.
3. The following class files have already been provided: **Form1.cs**, and **Minerals.cs**, arestored in the project folder. You will be programming methods in **Form1.cs** and **Minerals.cs**.
4. The **Form1.cs** class is a driver class that will load the (1) form, (2) gather user input via text box and a button, (3) create labels, (4) move the XY coordinates for all form objects, (5) create the **Minerals** objects base upon user input, and (6) handle all input errors. Listed below are its methods:
   1. ***Form1\_Load******( )***: when form starts anything in this method will be called first
   2. ***makeLabel( )***: used to create labels, assign XY coordinates and string information.
   3. ***userInputInt( ):*** handles user input for how many minerals to be created. Checks for errors.
   4. ***Next\_Click( ):*** method gathers controls user input to determine the next method call for the five steps through the program.
   5. ***makeMineralsProcess( )***: method is used to control the location of the text box **txtEntry** and button **btnEnterData,** and it also accepts a String object to be passed to ***makeLabel( )*** that will be in conjunction with the text box and button.
   6. ***userInputMineralSymbol( )***: method catches if the mineral symbol is beyond the parameters and assigns a value to variable for the constructor.
   7. ***userInputTempMineralName( )***: method is used to assign the mineral name to a variable for the constructor.
   8. ***userInputTempTonnage( )***: method is used to assign the mineral tonnage to a variable for the constructor. Checks of non-numerical entries.
   9. ***Enter\_KeyDown( )***: method allows user to use enter key.
5. The **Minerals.cs** object class will create the objects with the information for each mineral from the user input. Each object will need to store the mineral symbol, name, tonnage, and a market rate that will be randomly created. Listed below are its methods and constructors:
   1. ***Minerals( ):*** constructor that needs to take three arguments (String symbol, String name, and an integer for tonnage).
   2. ***setMarketRate( ):*** generates and assigns a value for the market rate which will be a random double value ranging from 1 to 200.
   3. ***ToString( ):*** method will create the print to screen information that will be used by **Form1** to print the results to the form.
   4. The Estimated Market Value of the mineral is calculated by the **market rate** **\*** **tonnage** for each mineral.
6. When interacting with the form, the user should be able to click the “Next” button or hit the enter key to input information from the text box with the same results. NOTE: the text “Next” for the **btnEnterData** button is created with code. The default text for the button is “EnterKey”.
7. When the user chooses to create 2 or more minerals, the program should have the user enter in all information (STEPS 2-4) before the program moves to STEP 5 printout.
8. Your code should attempt to automate the assignment of the XY coordinates of the objects as much as possible; however, you will have to hard code the XY coordinates, especially at the beginning.

**RUBRIC**

|  |  |  |
| --- | --- | --- |
| **Solution and Project (There is NO partial credit)** *(NOTE: UC represents uppercase and LC represents lowercase)* | | |
| The VS project file is present on the flash drive in a single folder with your contest ID |  | 20 points |
| **Program Execution (*If the program does not execute, then the remaining items in this section receive a score of zero)*** | | |
| When program starts user is prompted to enter in amount of minerals to create |  | 10 points |
| Button is renamed to “Next” |  | 10 points |
| STEP 1: |  |  |
| Program layout is similar to provided output example |  | 10 points |
| When entering information, the user can hit [ENTER-KEY] or they can click on “Next” button to proceed to next step |  | 20 points |
| Error message box appears if user attempts to enter in letters or symbols, and program does not crash |  | 30 points |
| Error message box appears if user attempts to enter in value beyond range (0 to 10), and program does not crash |  | 20 points |
| STEP 2: |  |  |
| Program layout is similar to provided output example with labels, text box, and button. Label “You selected to create ### minerals.” must have exact number entered. |  | 20 points |
| Error message box appears if user attempts to enter in value out of the range (3 to 5 values), and program does not crash |  | 20 points |
| STEP 3: |  |  |
| Program layout is similar to provided output example with labels, text box, and button. |  | 10 points |
| STEP 4: |  |  |
| Program layout is similar to provided output example with labels, text box, and button. |  | 20 points |
| Error message box appears if user attempts to enter in letters or symbols, and program does not crash |  | 30 points |
| STEP 5: |  |  |
| Final list of entered minerals appears |  | 30 points |
| Estimated Market Value has correct calculations and is formatted to US currency |  | 20 points |
| **Subtotal** |  | **/270 Points** |
| **Source Code Review** *NOTE: in order to get full credit you must place the comment flag in front of the comment in your code. The comment flag will precede the explanation. For example, if the flag is SC1, your comment must read as “#SC1…” in front of the part of the code being reviewed. Code must work to get credit.* |  |  |
| A comment containing the contestant number is present at the top of the Form1.cs file |  | 10 points |
| SC1:**Form1.cs** class: Place a SC1 comment by the code that programs the XY coordinates of the text box and button for STEP 1. |  | 20 points |
| SC2: **Form1.cs** class ***makeLabel( )***: method is used to create labels, allow autosizing, add text to labels, and place them at specific XY coordinates. |  | 30 points |
| SC3: **Form1.cs** class ***userInputInt( )***: method uses ***Try/Catch***  to stop wrong data entry for whole number inputs |  | 20 points |
| SC4: **Form1.cs** class ***userInputInt( )***: method detects if data is out of range (0 to 10) |  | 10 points |
| SC5: **Form1.cs** class ***Next\_Click( )***: method gathers controls user input to determine the next method call for the five steps through the program. |  | 30 points |
| SC6: **Form1.cs** class ***makeMineralsProcess( )***: method is used to control the location of the text box **txtEntry** and button **btnEnterData** |  | 20 points |
| SC7: **Form1.cs** class ***userInputMineralSymbol( )***: method catches if the mineral symbol is beyond the parameters; assigns value from text box to variable for constructor |  | 20 points |
| SC8: **Form1.cs** class ***userInputTempMineralName( )***: method is used to assign the mineral name to a variable for the constructor |  | 10 points |
| SC9: **Form1.cs** class ***userInputTempTonnage( )***: method is used to assign the mineral tonnage to a variable for the constructor |  | 10 points |
| SC10: **Form1.cs** class ***userInputTempTonnage( )***: method uses Try/Catch to stop non numerical entries |  | 20 points |
| SC11: **Form1.cs** class ***Enter\_KeyDown( )***: method allows for data inputted in text box to be submitted with enter key |  | 10 points |
| SC12: **Minerals.cs** class ***setMarketRate( )***: method uses Random object to create a double value in the range of 1 to 200. |  | 10 points |
| SC13: **Minerals.cs** class ***getValue( )***: method calculates estimates market. Value |  | 10 points |
| SC14: **Minerals.cs** class ***toString( )***: formats all dollar values to US currency. |  | 10 points |
| **Subtotal** |  | **/240 Points** |
| **Total Points** |  | **/510 Points** |

Machine generated alternative text:
...harp\Regional\C# Regional Grader\Conversion1\Form1.cs
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
using 
System;
using 
System.Collections.Generic;
using 
System.ComponentModel;
using 
System.Data;
using 
System.Drawing;
using 
System.Linq;
using 
System.Runtime.InteropServices;
using 
System.Text;
using 
System.Threading.Tasks;
using 
System.Windows.Forms;
using 
System.Windows.Forms.VisualStyles;
using static 
System.Windows.Forms.VisualStyles.
VisualStyleElement
;
using 
Button 
= System.Windows.Forms.
Button
;
using 
TextBox 
= System.Windows.Forms.
TextBox
;
namespace 
ChamaBasinMiner
{
public partial class 
MineralEntry 
: 
Form
{
int 
step = 0;
int 
MineralCount;
String 
MineralSymbol;
String 
tempMineralName;
int 
tempTonnage;
List
<
Minerals
> minerals = 
new 
List
<
Minerals
>();
int 
y = 60;
public 
MineralEntry
()
{
InitializeComponent();
}
private void 
Form1_Load(
object 
sender, 
EventArgs 
e)
{
//SC1
makeLabel(10, 20, 
"How many minerals types do you want to 
create?"
);
txtEntry.Location = 
new 
Point
(20, 40);
btnEnterData.Text = 
"Next"
;
btnEnterData.Location = 
new 
Point
(100, 40);
}
//SC2
private void 
makeLabel(
int 
x, 
int 
y, 
String 
str)
{
Label 
b = 
new 
Label
();
b.Location = 
new 
Point
(x, y)
;
b.AutoSize = 
true
;
b.Text = str
;
this
.Controls.Add(b);


Machine generated alternative text:
...harp\Regional\C# Regional Grader\Conversion1\Form1.cs
2
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
}
private void 
userInputInt(
TextBox 
t)
{
int 
value;
try   
//SC3
{
value = 
int
.Parse(t.Text);
if 
(value <= 0 || value > 10)  
//SC4
{
MessageBox
.Show(
"Your entry needs to be greater than 
zero and NOT greater than 10. Please enter a new 
number."
, 
"Error"
);
}
else
{
MineralCount = value;
makeLabel(10, 40, 
"You selected to create " 
+ value + 
" minerals."
);
makeLabel(10, 60, 
"You will now enter in the mineral 
symbol, its name and how much tonnage has been mined. " 
+ 
"The market rate for $ per ton will be randomly 
generated."
);
makeMineralsProcess(
"Please enter the three to five 
alphanumeric mineral symbol"
);
}
}
catch  
//SC3
{
MessageBox
.Show(
"You entered an inproper value. Please 
enter a whole number."
, 
"Error"
);
step
--
;
}
}
//SC5
private void 
Next_Click(
object 
sender, 
EventArgs 
e)
{
step++;
if 
(step == 1)
{
userInputInt(txtEntry);
}
if
(step == 2)


Machine generated alternative text:
...\Regional\Final Exam\C# Regional\Conversion1\Form1.cs
3
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
{
userInputMineralSymbol();
}
if
(step == 3)
{
userInputTempMineralName();
}
if
(step == 4)
{
userInputTempTonnage();
}
if 
(step == 5)
{
Minerals 
mineral = 
new 
Minerals
(MineralSymbol, 
tempMineralName, tempTonnage);
MineralSymbol = 
""
;
tempMineralName = 
""
;
tempTonnage = 0;
step = 1;
minerals.Add(mineral);
MineralCount
--
;
if
(MineralCount > 0)
{
makeMineralsProcess(
"Please enter the three to five 
alphanumeric mineral symbol"
);
}
else
{
for 
(
int 
q = 0; q < minerals.Count; q++)
{
y += 25;
makeLabel(10, y, minerals[q].toString());
}
}
}
}
//STUDENT POINT  //SC6
private void 
makeMineralsProcess(
String 
str)
{
if
(MineralCount > 0)
{
y += 20;
makeLabel(10, y, str);
y += 20;
txtEntry.Text = 
""
;
txtEntry.Location = 
new 
Point
(20, y); 
//SC6
btnEnterData.Location = 
new 
Point
(100, y); 
//SC6
}


Machine generated alternative text:
...\Regional\Final Exam\C# Regional\Conversion1\Form1.cs
4
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
else
{
for 
(
int 
q = 0; q < minerals.Count; q++)
{
minerals[q].toString();
}
}
}
//STUDENT POINT
private void 
userInputMineralSymbol()
{
MineralSymbol = txtEntry.Text;  
//SC7
if 
(MineralSymbol.Length < 3 || MineralSymbol.Length > 5)  
//
SC7
{
MessageBox
.Show(
"That is an improper value. Please Try 
Again."
, 
"Error"
);
step
--
;
}
else
{
makeMineralsProcess(
"Please enter the mineral name."
);
}
}
//STUDENT POINT
private void 
userInputTempMineralName()
{
tempMineralName = txtEntry.Text; 
//SC8
if
(txtEntry.Text 
is 
String
)
{
makeMineralsProcess(
"Please enter the total amount of tons
that was mined."
);
}
else
{
MessageBox
.Show(
"That is an improper value. Please Try 
Again."
, 
"Error"
);
step
--
;
}
}
//STUDENT POINT  
private void 
userInputTempTonnage()
{
String 
temp = txtEntry.Text;
try   
//SC10
{
tempTonnage = 
int
.Parse(temp); 
//SC9
}
catch
Machine generated alternative text:
...\Regional\Final Exam\C# Regional\Conversion1\Form1.cs
5
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
{
MessageBox
.Show(
"You entered an inproper value. Please 
enter a whole number."
, 
"Error"
);
step
--
;
}
}
private void 
Enter_KeyDown(
object 
sender, 
KeyEventArgs 
e)
{
Next_Click(sender, e);
//SC11
}
}
}


Machine generated alternative text:
...gional\Final Exam\C# Regional\Conversion1\Minerals.cs
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
using 
System;
using 
System.Drawing.Text;
namespace 
ChamaBasinMiner
{
internal class 
Minerals
{
String 
symbol;
String 
name;
double 
marketRate;
int 
tonnage;
int 
min = 1;
int 
max = 200;
public 
Minerals
()
{
symbol = 
""
;
name = 
""
;
marketRate = setMarketRate();
tonnage = 0;
}
public 
Minerals
(
String 
symbol, 
String 
name, 
int 
tonnage)
{
this
.symbol = symbol;
this
.name = name;
this
.tonnage = tonnage;
marketRate = setMarketRate();
}
//STUDENT POINT
private double 
setMarketRate()
{
Random 
r = 
new 
Random
();  
//SC12
marketRate = 
(
r.NextDouble() * (
(
max 
-
min
) 
+ 1) + min
)
;
return 
marketRate;
}
//STUDENT POINT
public double 
getValue()  
//SC13
{
double 
temp = marketRate * tonnage;
return 
temp;
}
//STUDENT POINT
public 
String 
toString()  
//SC14
{
return 
"[Mineral Symbol: " 
+ symbol + 
"].[" 
+ 
"Mineral Name: " 
+ name + 
"].[" 
+ 
" Market $/ton: $" 
+ 
String
.Format


Machine generated alternative text:
...gional\Final Exam\C# Regional\Conversion1\Minerals.cs
2
49
50
51
52
(
"
{
0
:
0.00
}
"
, marketRate) + 
"].[" 
+ 
" Total Tonnage Mined: " 
+
tonnage + 
"].[" 
+ 
"Estimated Market Value: $" 
+ 
String
.Format(
"
{
0
:
0.00
}
"
, 
this
.getValue())+
"]"
;
}
}
}
