Creative Software Design, Assignment 12-1

Deadline: 2024-11-13 23:59 (No score for late submission)

- Submit your homework by uploading your zip file to the LMS assignment section. Below is an example.

```
13178_Assignment1-1_2024123456.zip
├─ 1.cc
├─ 2.cc
├─ 3.cc
└─ ...
```

- Your zip file name should follow this format: **13178_Assignment[Assignment-number]_[Student-ID].zip**
  - Ex. 13178_Assignment1-1_2024123456.zip
- Source files should be named as **<filename>.cc** *or* **<filename>.cpp**
- **You must submit your solution in the zip file before the deadline.**

1. **Write a C++ program that implements the Exp function to calculate powers with exception handling for negative exponents.**

   A. **`Exp` function:**

      1. Implement `Exp` with two int parameters: `base` and `exponent`.
      2. If `exponent` is negative, `Exp` should throw an exception.

   B. **Exception Handling**

      1. Use a `try` block to call `Exp` and a `catch` block to handle exceptions.

   C. **Example of the `main()` function**

```
int main() {
    int v = 0;
    try {
        v = Exp(2, 10);
        cout << "The value of 2 to the power 10 is " << v << endl;

        v = Exp(2, -10);
        cout << "The value of 2 to the power -10 is " << v << endl;
    }
    catch(const char *s) {
        cout << "Exception! " << s << endl;
    }
}
```

   D. Example output of your program (Bold text indicates user input):

```
The value of 2 to the power 10 is 1024
Exception! Cannot use negative numbers.
```

   E. Submission file: one C++ source file (File name: 1.**cc** or 1.**cpp**)

2. **Write a divide() function in C++ that performs division with exception handling for invalid inputs and division by zero.**

   A. **Define `divide()` Function and Exception Classes:**

      1. Define an `Exceptions` base class.
      2. Define `InvalidInputException` and `DivideByZeroException` classes that inherit from `Exceptions`.
      3. Implement the `divide()` function to:

         i. Throw `InvalidInputException` if either parameter is negative.

         ii. Throw `DivideByZeroException` if division by zero is attempted.

   B. **Exception Handling**

      1. Use a `try` block in `main()` to call `divide()`, and `catch` blocks to handle each exception type by calling `e.print()`.

   C. **Example of the `main()` function:**

```
int main() {
    int x, y;
    double result;
    try {
        cout << "Division. Input two numbers >> ";
        cin >> x >> y;

        result = divide(x, y);
        cout << result << endl;
    }
    catch(DivideByZeroException &e) {
        e.print();
    }
    catch(InvalidInputException &e) {
        e.print();
    }
}
```

   D. Example output of your program (Bold text indicates user input):

```
Division. Input two numbers >> 1000 8
125

Division. Input two numbers >> 10 -10
Negative value input exception

Division. Input two numbers >> 10 0
Divide by zero exception
```

   E. Submission file: one C++ source file (File name: 2.**cc** or 2.**cpp**)

3. **Modify the program from problem 2 of Assignment 8-1 to handle both positive and negative inputs.**

A. **Modify `Converter` and `DollarToWon` Classes**

1. **`Converter` Class:**

   i. Keep the `Converter` class structure as is, but ensure the `run()` method throws an exception if a negative input is detected.

2. **`DollarToWon` Class:**

   i. Inherit from Converter and implement the `convert()`, `getSrcMetric()`, and `getDestMetric()` functions.

   ii. Ensure that `convert()` handles only positive `src` values, throwing an exception if a negative value is passed.

B. **Exception Handling**

1. Modify `run()` in Converter to use exception handling when negative inputs are detected.

C. **Original code from Assignment 8-1, Q2:**

```
class Converter {
protected:
   double _ratio;
   virtual double convert(double src) = 0;
   virtual string getSrcMetric() = 0;
   virtual string getDestMetric() = 0;

public:
   Converter(double ratio) : _ratio(ratio) { }
   void run() {
      double src;
      cout << "Convert " << getSrcMetric() << " to " << getDestMetric() << endl;
      cout << "Input " << getSrcMetric() << " : ";
      cin >> src;
      cout << "Result : " << convert(src) << getDestMetric() << endl;
   }
};

class DollarToWon : public Converter {
public:
   DollarToWon(double ratio = 0.0) : Converter(ratio) {}
   double convert(double src) { return src * _ratio; }
   string getSrcMetric() { return " dollar"; }
   string getDestMetric() { return " won"; }
};

int main() {
   DollarToWon dtw(1176.5);
   dtw.run();
}
```

D. Example output of your program (Bold text indicates user input):

```
Convert dollar to won
Input dollar : 5
Result : 5882.5 won

Convert dollar to won
Input dollar : -10
Exception! Cannot convert negative value
```

E. Submission file: one C++ source file (File name: 3.**cc** or 3.**cpp**)