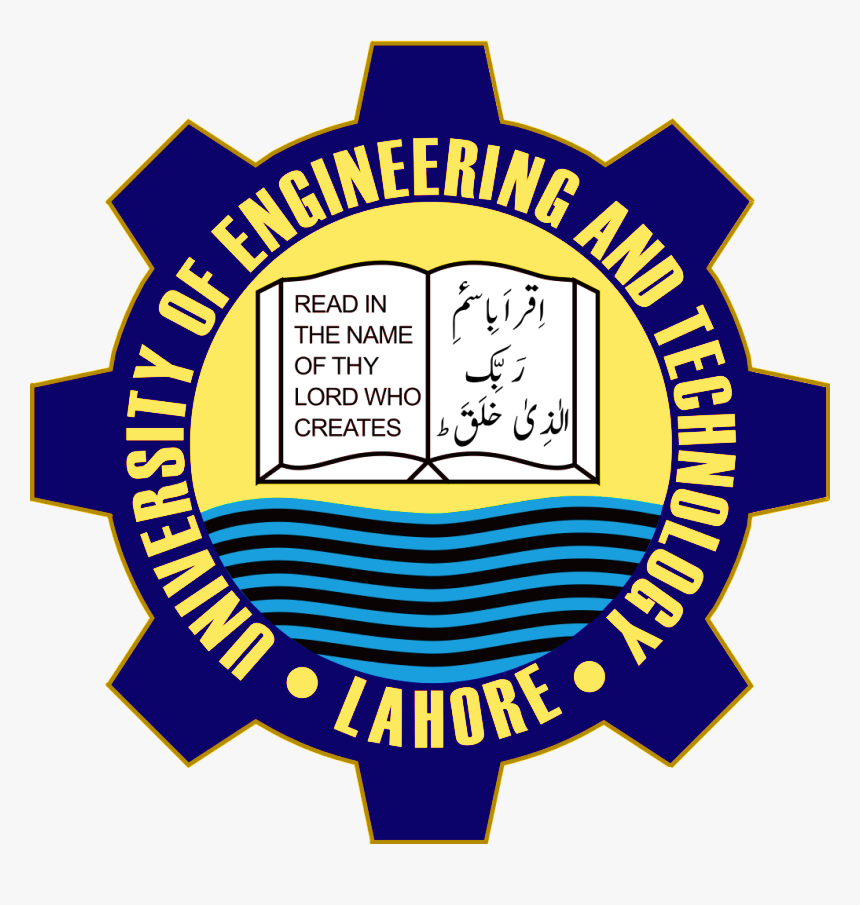
**Rock, Paper, Scissor OOP Project**



**Session: 2021**

**Submitted by:**

**Wali Muhammad 2021-SE-39**

**Submitted to:**

**Ma’am Zahra**

Department of Computer Science, New Campus **University of Engineering and Technology Lahore, Pakistan**

**Introduction**

In real life, Rock Paper Scissor is a hand game usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are “Rock” “Paper” and “Scissors”. This Project is based on this game.

**Project Description**

Implement a class called Tool. It should have an int field called strength and a char field called type. You may make them either private or protected. The Tool class should also contain the function void setStrength(int), which sets the strength for the Tool.

Create 3 more classes called Rock, Paper, and Scissors, which inherit from Tool. Each of these classes will need a constructor to take in an int that is used to initialise the strength field. The constructor should also initialize the type field using 'r' for Rock, 'p' for Paper, and 's' for Scissors.

These classes will also need a public function bool fight(Tool) that compares their strengths in the following way:

* Rock's strength is doubled (temporarily) when fighting scissors but halved (temporarily) when fighting paper.
* In the same way, paper has the advantage over rock and scissors against the paper.
* The strength field shouldn't change in the function, which returns true if the original class wins in strength and false otherwise.

You may also include any extra auxiliary functions and/or fields in any of these classes. Run the program without changing the main function.

**Methodologies**

This section will cover the following concepts:

* Classes
* Objects
* Encapsulation
* Inheritance
* Polymorphism

**Classes**

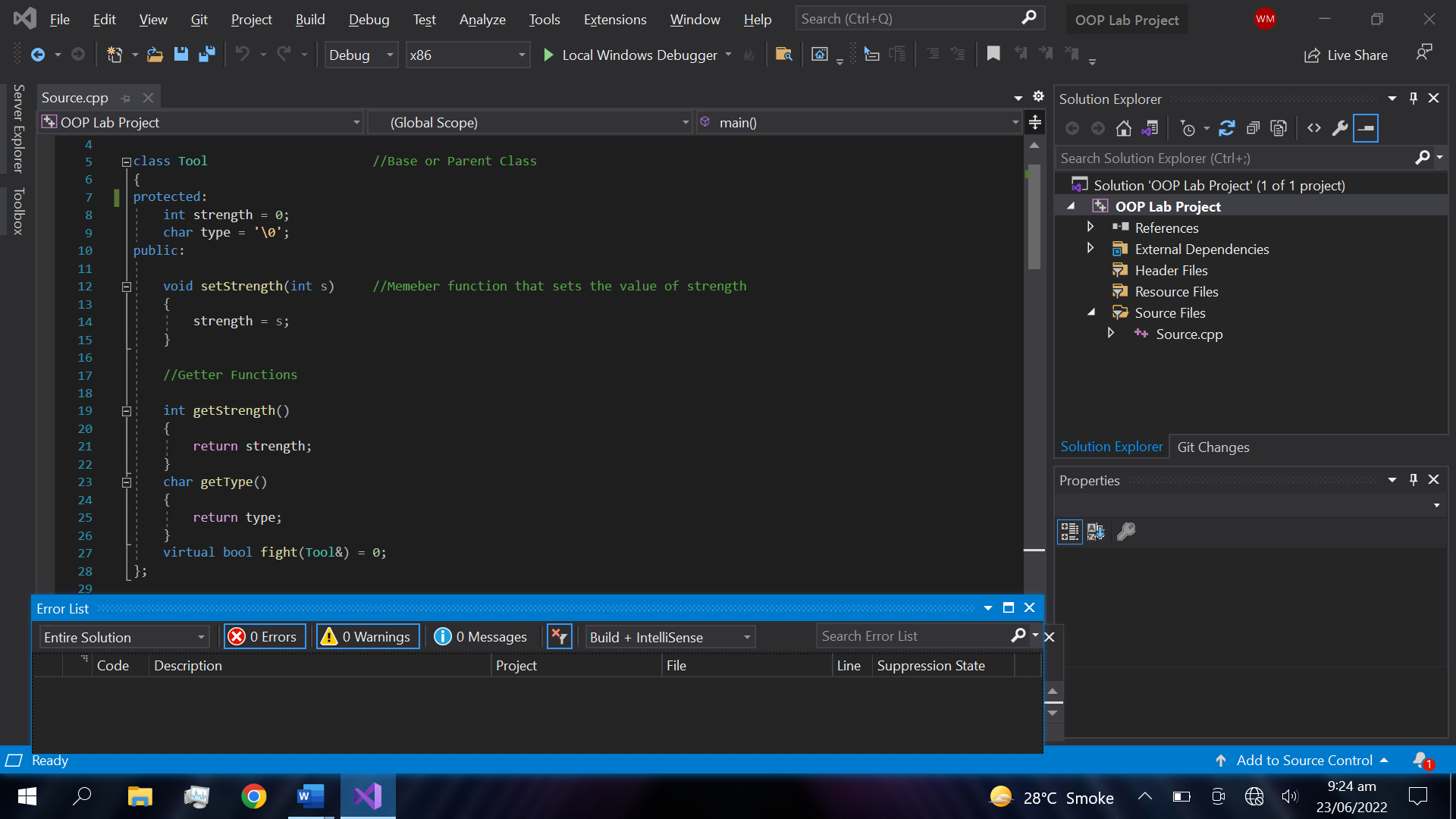
The classes mentioned in the Project statement are implemented. There are total 4 classes.

## Base Class

* Tool

It is an Abstract class. It has an int field called strength and a char field called type. The access specifier of Tool class variables is protected. The Tool class contain the function void setStrength(int), with an integer parameter, which sets the strength for the Tool.

There are two getter functions that gets the value of attributes. It has a bool fight which is a Pure Virtual function that accepts a reference object.



## Derived Classes

There are three derived classes.

* Paper
* Rock
* Scissor

Each of these classes has a constructor which takes an int that is used to initialize the strength field. The constructor also initializes the type field using 'r' for Rock, 'p' for Paper, and 's' for Scissors. These classes also have a public function bool fight(Tool) that compares their strengths.

**Objects**

There are 3 objects of child classes (Rock, Paper, and Scissors) in main function given below:

Rock r1(15);

Paper p1(7);

Scissors s1(5);

The values of there attribute Strength are passed in Constructor that initialize the strength field with these values. These classes have a public function bool fight(Tool) that compares their strengths in the following way:

* Rock's strength is doubled (temporarily) when fighting scissors, but halved (temporarily) when fighting paper and vice versa.
* The strength field shouldn't change in the function, which returns true if the original class wins in strength and false otherwise.

**Encapsulation**

The parent class attributes are protected. So, these are not directly accessible in main function. The fight function contains all the if-else conditions that compares the strength of rock, paper, scissors strength attribute. All the if-else conditions are encapsulated in fight function and we are calling the fight function which is executed several times in main function, that executes all the conditional statement.

**Inheritance**

Rock, Paper, and Scissors classes are inherited from an Abstract class Tool. There access specifiers are public so these derived classes have access on parent class attributes.

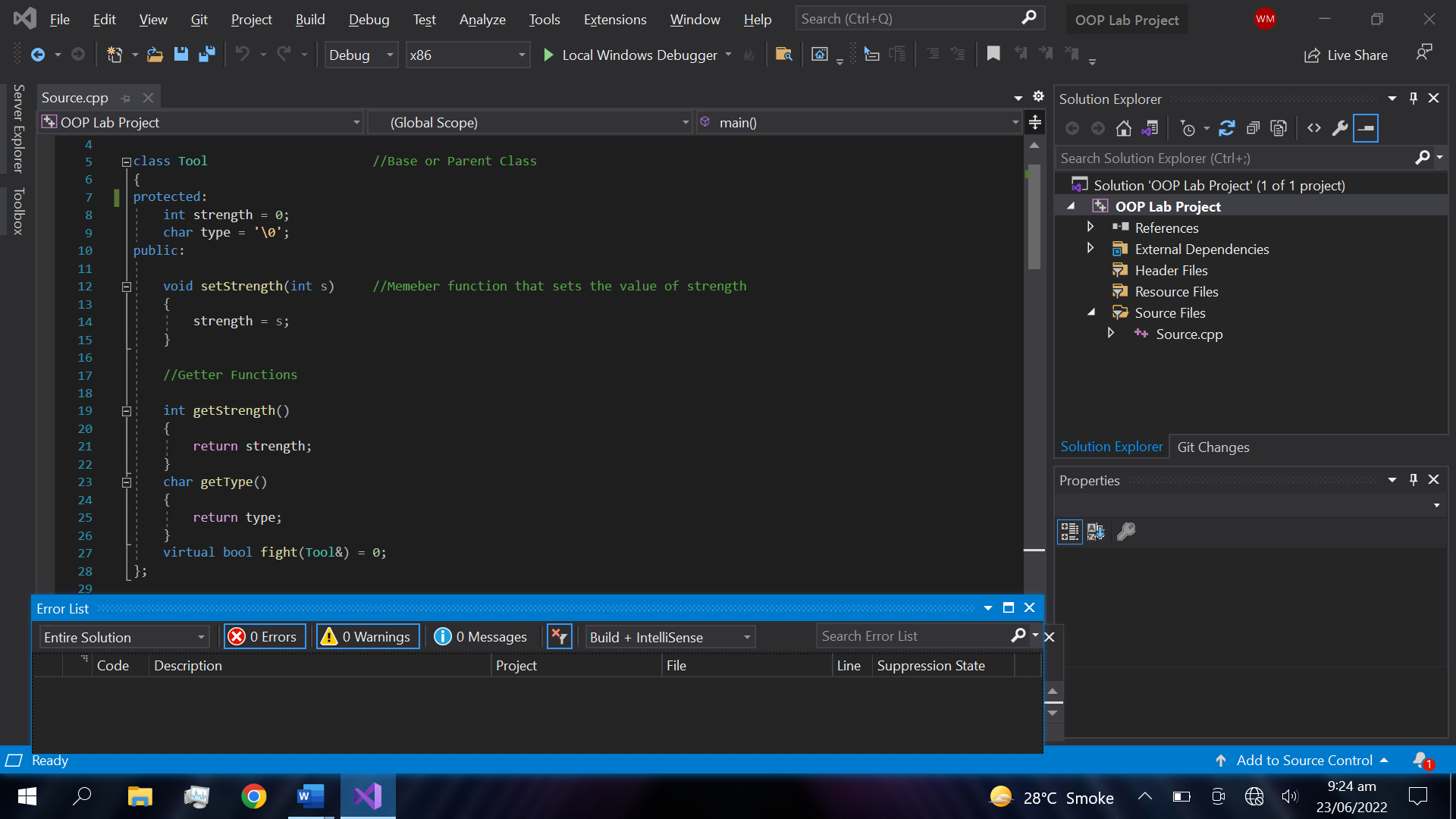
class Rock : public Tool

class Paper : public Tool

class Scissors : public Tool

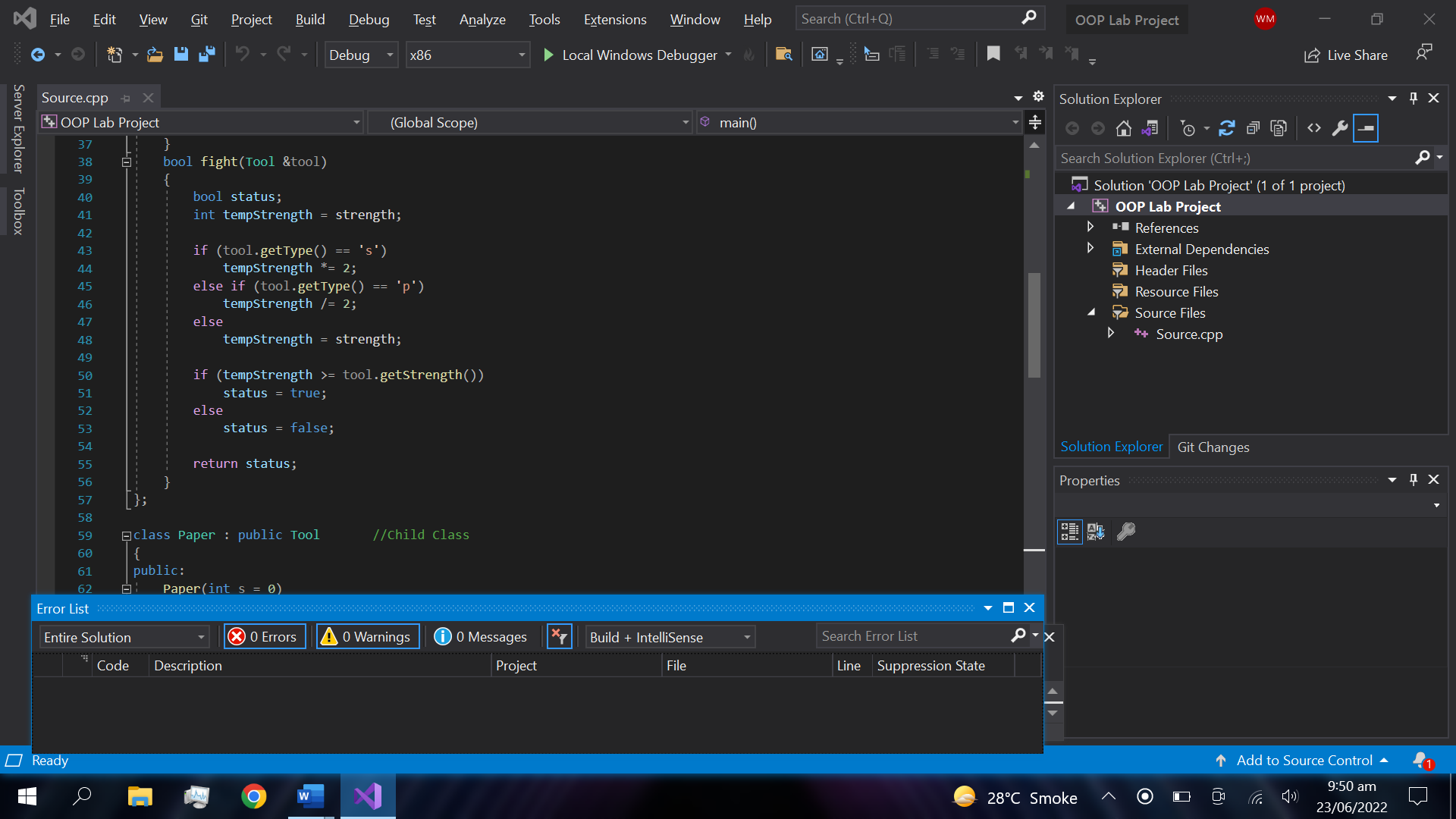
**Polymorphism**

The Tool class have a Pure Virtual Function that makes it an Abstract Class. This function accepts a Reference object that compare the values of strength of objects.

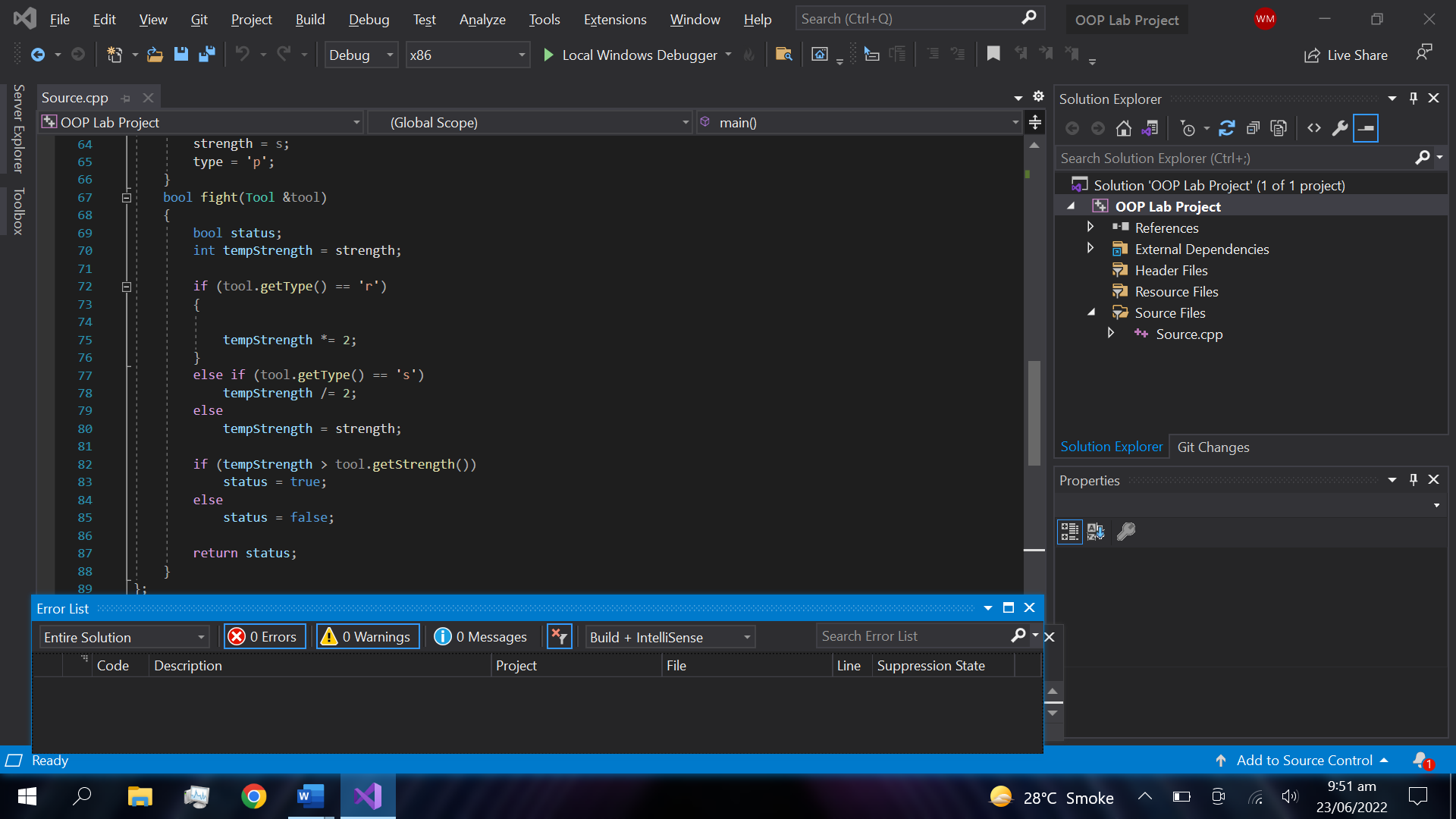
****

Pure Virtual function is inherited in all the child classes and all the child classes have it own definition. The return type of this function is Bool. This function returns a bool value which is either true or false, depends upon the derived class strength.

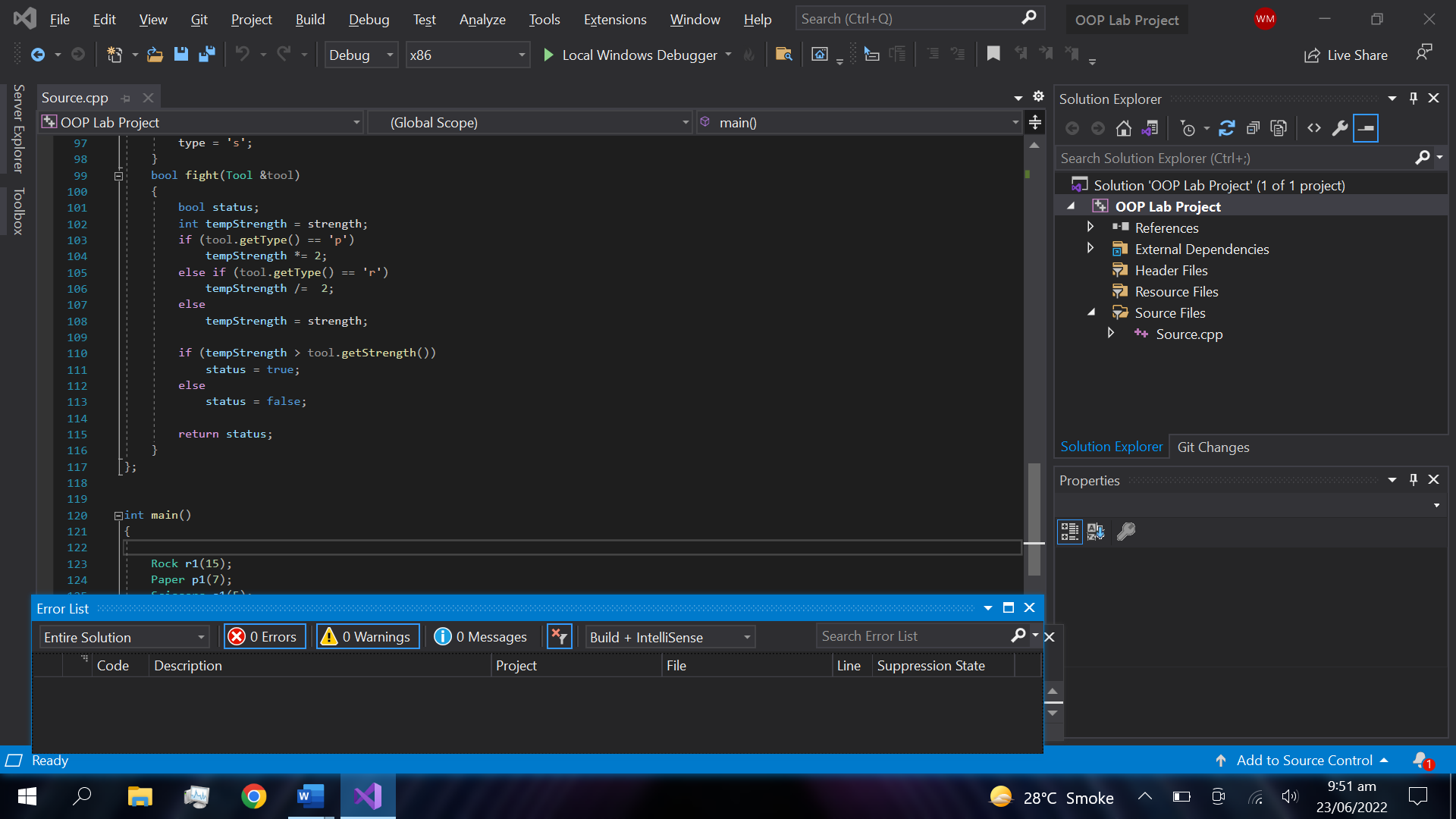
**Rock Class Fight Function Defination**



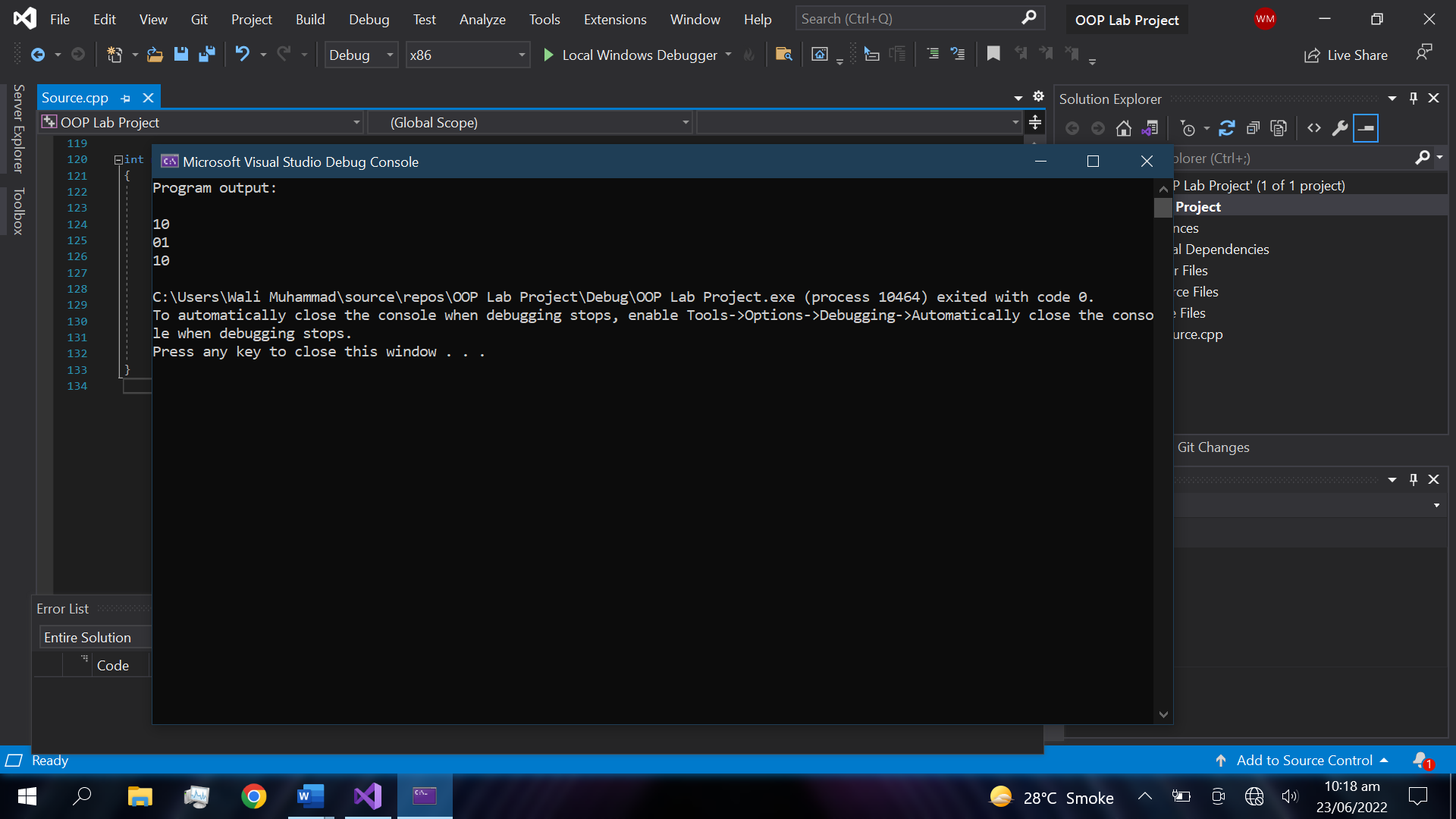
**Paper Class Fight Function Defination**



**Scissor Class Fight Function Defination**

****

**Programs Output**

****

* **1 Means true or the object wins**
* **0 Means false of the object lose**

**Code**

#include <iostream>

#include<string>

using namespace std;

class Tool //Base or Parent Class

{

public:

int strength = 0;

char type = '\0';

public:

void setStrength(int s) //Memeber function that sets the value of strength

{

strength = s;

}

int getStrength()

{

return strength;

}

char getType()

{

return type;

}

virtual bool fight(Tool&) = 0;

};

class Rock : public Tool //Child Class

{

public:

Rock(int s = 0)

{

strength = s;

type = 'r';

}

bool fight(Tool &tool)

{

bool status;

int tempStrength = strength;

if (tool.getType() == 's')

tempStrength \*= 2;

else if (tool.getType() == 'p')

tempStrength /= 2;

else

tempStrength = strength;

if (tempStrength >= tool.getStrength())

status = true;

else

status = false;

return status;

}

};

class Paper : public Tool //Child Class

{

public:

Paper(int s = 0)

{

strength = s;

type = 'p';

}

bool fight(Tool &tool)

{

bool status;

int tempStrength = strength;

if (tool.getType() == 'r')

{

tempStrength \*= 2;

}

else if (tool.getType() == 's')

tempStrength /= 2;

else

tempStrength = strength;

if (tempStrength > tool.getStrength())

status = true;

else

status = false;

return status;

}

};

class Scissors : public Tool //Child Class

{

public:

Scissors(int s = 0)

{

strength = s;

type = 's';

}

bool fight(Tool &tool)

{

bool status;

int tempStrength = strength;

if (tool.getType() == 'p')

tempStrength \*= 2;

else if (tool.getType() == 'r')

tempStrength /= 2;

else

tempStrength = strength;

if (tempStrength > tool.getStrength())

status = true;

else

status = false;

return status;

}

};

int main()

{

Rock r1(15);

Paper p1(7);

Scissors s1(5);

cout << "Program output: \t\t\t\n" << endl;

cout << s1.fight(p1) << p1.fight(s1) << endl;

cout << p1.fight(r1) << r1.fight(p1) << endl;

cout << r1.fight(s1) << s1.fight(r1) << endl;

return 0;

}

**The End**