Template classes

```
template<class T>
class Arithmetic {
 private:
   T a;
   T b;
 public:
   Arithmetic(T a, T b);
   T add();
   T sub();
 };

template<class T>
Arithmetic<T>:: Arithmetic(T a, T b){
     this->a =a;
     this->b =b;
template <class T>
T Arithmetic :: add(){
   T c;
   c = a+b;
   return c;
}
```

not always use T, use when needed

```
template <class T>
int arithmetic<T> :: sub() {
    T c;
    c = a - b;
    return c;
}

int main() {
  Arithmetic<int> ar(10, 5);
  cout << ar.add();
  Arithmetic<float> ar1(1.5, 1.2);
  cout << ar1.add();
}
```

everything will be int

everything will be float