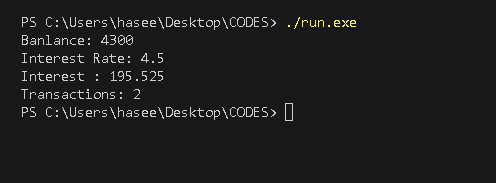
**Task 1:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Bank  {  private:      double balance;      double interestRate;      double interest;      int transaction;  public:      Bank() : balance(0), interestRate(4.5), interest(0), transaction(0) {}      Bank(double bal, double interest) : balance(bal), interestRate(interest), interest(0), transaction(0) {}      Bank &setInterestRate(double intrate)      {          interestRate = intrate;          return \*this;      }      Bank &makeDeposit(double deposite)      {          balance += deposite;          calcInterest();          inCount();          return \*this;      }      Bank &withdraw(double witdraw)      {          if (balance > witdraw)          {              inCount();              balance -= witdraw;          }          else          {              cout << "Insufficent balance." << endl;          }          return \*this;      }      Bank &calcInterest()      {          interest = (balance \* (4.5 / 100));          return \*this;      }      void inCount() { ++transaction; }      double getCount() { return transaction; }      double getInterestRate() { return interestRate; }      double getInterest() { return interest; }      double getTransactions() { return transaction; }      double getBalance() const { return balance; }      void display()      {          cout << "Banlance: " << getBalance() << endl;          cout << "Interest Rate: " << getInterestRate() << endl;          cout << "Interest : " << getInterest() << endl;          cout << "Transactions: " << getTransactions() << endl;      }  };  int main()  {      Bank a;      a.makeDeposit(4345).calcInterest().withdraw(45).display();      return 0;  } |

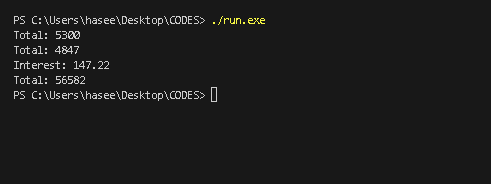
**Output:**

****

**Task 2:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Account  {      double balance;  public:      Account(double b = 0)      {          if (b < 0)          {              cout << "balance cannot be negative." << endl;              balance = 0;          }          else          {              balance = b;          }      }      void credit(double bal)      {          balance += bal;      }      bool debit(double bal)      {          if (bal > balance)          {              cout << "Insufficent balance." << endl;              return false;          }          else          {              balance -= bal;              return true;          }      }      double getBalance()      {          return balance;      }  };  class SavingsAccount : public Account  {  private:      double interestRate;  public:      SavingsAccount(double bal, double intrest) : Account(bal), interestRate(intrest) {}      double calculateInterest()      {          double interest = interestRate \* getBalance() / 100;          return interest;      }  };  class CheckingAccount : public Account  {      double transactionFee;  public:      CheckingAccount(double bal, double transtfee) : Account(bal), transactionFee(transtfee) {}      void credit(double bal)      {          Account ::credit(bal - transactionFee);      }      bool debit(double bal)      {          if (Account::debit(bal))          {              double balacccccc = getBalance();              balacccccc -= transactionFee;              return true;          }          else          {              return false;          }      }  };  int main()  {      Account A1(5000);      A1.credit(300);      cout << "Total: " << A1.getBalance() << endl;      A1.debit(453);      cout << "Total: " << A1.getBalance() << endl;      SavingsAccount sacount(34, 433);      cout << "Interest: " << sacount.calculateInterest() << endl;      CheckingAccount check(56325, 43);      check.credit(345);      check.debit(45);      cout << "Total: " << check.getBalance() << endl;      return 0;  } |

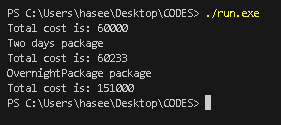
**Output:**

****

**Task 3:**

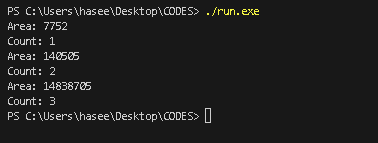
|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  class Package  {  protected: // Change to protected for derived classes to access      string sender\_name, sender\_address, sender\_city, sender\_state, sender\_Zip;      string recipient\_name, recipient\_address, recipient\_city, recipient\_state, recipient\_zip;      double weight, cost\_per\_ounce;  public:      Package(string send\_name, string send\_address, string send\_city, string send\_state, string send\_Zip, string recipt\_name, string recipt\_address, string recipt\_city, string recipt\_state, string recipt\_zip, double w, double cos) : sender\_name(send\_name), sender\_address(send\_address), sender\_city(send\_city), sender\_state(send\_state), sender\_Zip(send\_Zip), recipient\_name(recipt\_name), recipient\_address(recipt\_address), recipient\_city(recipt\_city), recipient\_state(recipt\_state), recipient\_zip(recipt\_zip), weight(w), cost\_per\_ounce(cos)      {          if (weight < 0 || cost\_per\_ounce < 0)          {              cout << "Error: Weight and cost per ounce must be positive." << endl;          }      }      double calculateCost()      {          return weight \* cost\_per\_ounce;      }  };  class TwoDayPackage : public Package  {      double todays\_fee;  public:      TwoDayPackage(string send\_name, string send\_address, string send\_city, string send\_state, string send\_Zip, string recipt\_name, string recipt\_address, string recipt\_city, string recipt\_state, string recipt\_zip, double w, double cos, double add\_fee) : Package(send\_name, send\_address, send\_city, send\_state, send\_Zip, recipt\_name, recipt\_address, recipt\_city, recipt\_state, recipt\_zip, w, cos), todays\_fee(add\_fee) {}      double calculateCost()      {          return Package::calculateCost() + todays\_fee;      }  };  class OvernightPackage : public Package  {      double additional\_fee\_per\_ounce;  public:      OvernightPackage(string send\_name, string send\_address, string send\_city, string send\_state, string send\_Zip, string recipt\_name, string recipt\_address, string recipt\_city, string recipt\_state, string recipt\_zip, double w, double cos, double additional\_fee\_per\_ounce) : Package(send\_name, send\_address, send\_city, send\_state, send\_Zip, recipt\_name, recipt\_address, recipt\_city, recipt\_state, recipt\_zip, w, cos), additional\_fee\_per\_ounce(additional\_fee\_per\_ounce) {}      double calculateCost()      {          return Package::calculateCost() + (additional\_fee\_per\_ounce \* weight);      }  };  int main()  {      Package parsal1 = Package("Haseeb", "Zarghonabad", "Quetta", "Pakistan", "5730", "Ahemd", "johar", "Lahore", "Paksitan", "87500", 200, 300);      cout << "Total cost is: " << parsal1.calculateCost() << endl;      cout << "Two days package" << endl;      TwoDayPackage tt("Haseeb", "Zarghonabad", "Quetta", "Pakistan", "5730", "Ahemd", "johar", "Lahore", "Paksitan", "87500", 200, 300, 233);      cout << "Total cost is: " << tt.calculateCost() << endl;      OvernightPackage yy("Haseeb", "Zarghonabad", "Quetta", "Pakistan", "5730", "Ahemd", "johar", "Lahore", "Paksitan", "87500", 200, 300, 455);      cout << "OvernightPackage package" << endl;      cout << "Total cost is: " << yy.calculateCost() << endl;        return 0;  } |

**Output:**

**  
Task 4**

|  |
| --- |
| #include <iostream>  using namespace std;  class Area {  private:      int length;      int width;      int area;      static int count;  public:      Area(int l, int w) {          this->length = l;          this->width = w;          this->area = this->length \* this->width;          count++;      }      static int getCount() {          return count;      }      void display() {          cout << "Area: " << this->area << endl;          cout << "Count: " << this->getCount() << endl;      }  };  int Area::count = 0;  int main() {      Area yy(24, 323);      yy.display();      Area ee(435, 323);      ee.display();      Area tt(4335, 3423);      tt.display();      return 0;  } |

**Output:**

****

**Task 5**

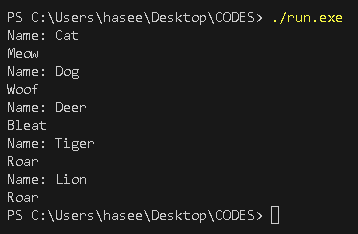
|  |
| --- |
|  |

**Output:**

**Task 6:**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  class Animal  {  protected:      string name;  public:      Animal(string name) : name(name) {}      void sound()      {          cout << "Name: " << name << endl;          cout << "Animals." << endl;      }  };  class Cat : public Animal  {  public:      Cat(string name) : Animal(name) {}      void sound()      {          cout << "Name: " << name << endl;          cout << "Meow" << endl;      }  };  class Dog : public Animal  {  public:      Dog(string name) : Animal(name) {}      void sound()      {          cout << "Name: " << name << endl;          cout << "Woof" << endl;      }  };  class TigerFamily : public Animal  {  public:      TigerFamily(string name) : Animal(name) {}  };  class Deer : public Animal  {  public:      Deer(string name) : Animal(name) {}      void sound()      {          cout << "Name: " << name << endl;          cout << "Bleat" << endl;      }  };  class Tiger : public TigerFamily  {  public:      Tiger(string name) : TigerFamily(name) {}      void sound()      {          cout << "Name: " << name << endl;          cout << "Roar" << endl;      }  };  class Lion : public TigerFamily  {  public:      Lion(string name) : TigerFamily(name) {}      void sound()      {          cout << "Name: " << name << endl;          cout << "Roar" << endl;      }  };  int main()  {      Cat cat("Cat");      cat.sound();      Dog dog("Dog");      dog.sound();      Deer deer("Deer");      deer.sound();      Tiger tiger("Tiger");      tiger.sound();      Lion lion("Lion");      lion.sound();      return 0;  } |

**Output:**

****