**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №4**

Выполнил: студент группы ПИ-3-21

Копжашаров Азамат

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

Для участия в ежегодном трансконтинентальном Ралли-марафоне «Дакар» по усложненной трассе в Южной Америке выбраны грузовики КАМАЗ и Tatra, которые после 2000 года лидируют в зачете грузовиков.

Решите задачу сравнения скоростей движения грузовиков по разным по проходимости участкам трассы, а именно: по равнине, горам, пустыне. Создайте и выдайте на экран таблицу результатов ралли - марафона. Определите победителя.

Для решения задачи используйте классы **Kamaz** и **Tatra,** а такжефункциюсравнения скоростей **FrCreater (**). Функция  **FrCreater (**) возвращает число +1, если объект **kamaz** движется быстрее объекта **tatra**; нуль, если их скорости одинаковы; число -1, если объект **kamaz** движется медленнее объекта **tatra** .

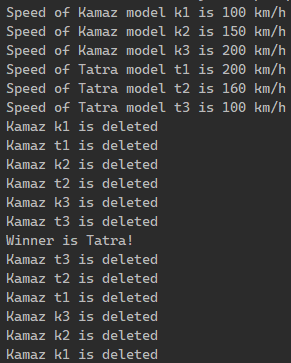
Оба класса содержат поля: «скорость» и «наименование» грузовой машины, а также методы: инициализация и отображение полей на экране. Определитесь с идентификаторами доступа к членам класса, не нарушая принципа инкапсуляции.

**Исходные данные:**

**Исходный код программы на C++:**

/// 4.1  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
class Kamaz {  
private:  
 int speed;  
 string name;  
public:  
 Kamaz(int givenSpeed, const string& givenName): speed(givenSpeed), name(givenName){}  
 ~Kamaz(){  
 cout << "Kamaz " << name << " is deleted" << endl;  
 }  
 void display(){  
 cout << "Speed of Kamaz model " << name << " is " << speed << " km/h" << endl;  
 }  
 int getSpeed(){  
 return speed;  
 }  
 string getName(){  
 return name;  
 }  
};  
  
class Tatra {  
private:  
 int speed;  
 string name;  
public:  
 Tatra(int givenSpeed, const string& givenName): speed(givenSpeed), name(givenName){}  
 ~Tatra(){  
 cout << "Kamaz " << name << " is deleted" << endl;  
 }  
 void display(){  
 cout << "Speed of Tatra model " << name << " is " << speed << " km/h" << endl;  
 }  
 int getSpeed(){  
 return speed;  
 }  
 string getName(){  
 return name;  
 }  
};  
  
int FrCreater(Kamaz k, Tatra t){  
 if(k.getSpeed() > t.getSpeed()) return 1;  
 else if(k.getSpeed() == t.getSpeed()) return 0;  
 else return -1;  
}  
  
int main() {  
  
 Kamaz k1(100, "k1"), k2(150, "k2"), k3(200, "k3");  
 Tatra t1(200, "t1"), t2(160, "t2"), t3(100, "t3");  
  
 k1.display();  
 k2.display();  
 k3.display();  
 t1.display();  
 t2.display();  
 t3.display();  
  
 int result1 = FrCreater(k1, t1), result2 = FrCreater(k2, t2), result3 = FrCreater(k3, t3);  
 if((result1 + result2 + result3) > 0) cout << "Winner is Kamaz!" << endl;  
 else if((result1 + result2 + result3) < 0) cout << "Winner is Tatra!" << endl;  
 else cout << "Draw!" << endl;  
  
 return 0;  
}

**Тесты:**

****

# **Задание №2**

**Постановка задачи:**

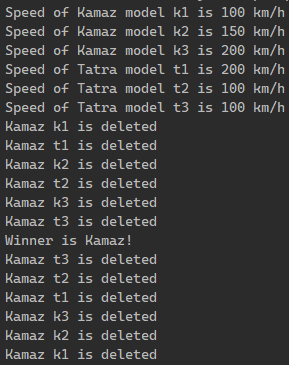
Сделайте функцию **FrCreater(**) дружественной: классу **Kamaz,** обоим классам **Kamaz** и **Tatra.**

**Исходные данные:**

**Исходный код программы на C++:**

// 4.2  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
class Tatra;  
  
class Kamaz {  
private:  
 int speed;  
 string name;  
public:  
 Kamaz(int givenSpeed, const string& givenName): speed(givenSpeed), name(givenName){}  
 ~Kamaz(){  
 cout << "Kamaz " << name << " is deleted" << endl;  
 }  
 void display(){  
 cout << "Speed of Kamaz model " << name << " is " << speed << " km/h" << endl;  
 }  
 int getSpeed(){  
 return speed;  
 }  
 string getName(){  
 return name;  
 }  
 friend int FrCreater(Kamaz k, Tatra t);  
};  
  
class Tatra {  
private:  
 int speed;  
 string name;  
public:  
 Tatra(int givenSpeed, const string& givenName): speed(givenSpeed), name(givenName){}  
 ~Tatra(){  
 cout << "Kamaz " << name << " is deleted" << endl;  
 }  
 void display(){  
 cout << "Speed of Tatra model " << name << " is " << speed << " km/h" << endl;  
 }  
 int getSpeed(){  
 return speed;  
 }  
 string getName(){  
 return name;  
 }  
 friend int FrCreater(Kamaz k, Tatra t);  
};  
  
int FrCreater(Kamaz k, Tatra t){  
 if(k.speed > t.speed) return 1;  
 else if(k.speed == t.speed) return 0;  
 else return -1;  
}  
  
int main() {  
  
 Kamaz k1(100, "k1"), k2(150, "k2"), k3(200, "k3");  
 Tatra t1(200, "t1"), t2(100, "t2"), t3(100, "t3");  
  
 k1.display();  
 k2.display();  
 k3.display();  
 t1.display();  
 t2.display();  
 t3.display();  
  
 int result1 = FrCreater(k1, t1), result2 = FrCreater(k2, t2), result3 = FrCreater(k3, t3);  
 if((result1 + result2 + result3) > 0) cout << "Winner is Kamaz!" << endl;  
 else if((result1 + result2 + result3) < 0) cout << "Winner is Tatra!" << endl;  
 else cout << "Draw!" << endl;  
  
 return 0;  
}

**Тесты:**



# **Задание №3**

**Постановка задачи:**

**Опишите класс, реализующий стек (Stack).**

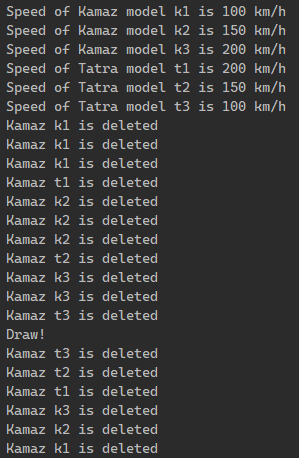
Сделайте класс **Tatra** дружественным классу **Kamaz.**

**Исходные данные:**

**Исходный код программы на C++:**

// 4.3  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
class Tatra;  
  
class Kamaz {  
private:  
 int speed;  
 string name;  
public:  
 Kamaz(int givenSpeed, const string& givenName): speed(givenSpeed), name(givenName){}  
 ~Kamaz(){  
 cout << "Kamaz " << name << " is deleted" << endl;  
 }  
 void display(){  
 cout << "Speed of Kamaz model " << name << " is " << speed << " km/h" << endl;  
 }  
 int getSpeed(){  
 return speed;  
 }  
 string getName(){  
 return name;  
 }  
 friend class Tatra;  
};  
  
class Tatra {  
private:  
 int speed;  
 string name;  
public:  
 Tatra(int givenSpeed, const string& givenName): speed(givenSpeed), name(givenName){}  
 ~Tatra(){  
 cout << "Kamaz " << name << " is deleted" << endl;  
 }  
 void display(){  
 cout << "Speed of Tatra model " << name << " is " << speed << " km/h" << endl;  
 }  
 int getSpeed(){  
 return speed;  
 }  
 string getName(){  
 return name;  
 }  
 int decide(Kamaz k){  
 if(k.speed > speed) return 1;  
 else if(k.speed < speed) return -1;  
 else return 0;  
 }  
};  
  
int FrCreater(Kamaz k, Tatra t){  
 if(t.decide(k) == 1) return 1;  
 else if(t.decide(k) == 0) return 0;  
 else return -1;  
}  
  
int main() {  
  
 Kamaz k1(100, "k1"), k2(150, "k2"), k3(200, "k3");  
 Tatra t1(200, "t1"), t2(150, "t2"), t3(100, "t3");  
  
 k1.display();  
 k2.display();  
 k3.display();  
 t1.display();  
 t2.display();  
 t3.display();  
  
 int result1 = FrCreater(k1, t1), result2 = FrCreater(k2, t2), result3 = FrCreater(k3, t3);  
 if((result1 + result2 + result3) > 0) cout << "Winner is Kamaz!" << endl;  
 else if((result1 + result2 + result3) < 0) cout << "Winner is Tatra!" << endl;  
 else cout << "Draw!" << endl;  
  
 return 0;  
}

**Тесты:**



# **Ответы на вопросы:**

**1. Что такое дружественная функция, для чего она нужна?**

**Дружественная функция**—это функция, не являющаяся членом класса, но обеспечивающая доступ ко всем членам класса, для которого она дружественна. Независима от класса **дружественная функция** находится вне класса. Но она объявляется внутри класса как обычная функция и перед ней указывается ключевое слово **friend**. Вне класса перед описанием дружественной функции это ключевое слово не ставится**.**

**2. Как она объявляется, каков её синтаксис?**

Синтаксис прототипа дружественной функции

***friend*** тип имя ([список\_ параметров])

**3. Через что дружественная функция может получить доступ к закрытому полю класса?**

Так как дружественная функция не является методом класса, для которого она дружественна, нельзя вызвать дружественную функцию обычным способом в точечном формате с помощью команды **a.show()**, а можно только командой **show(a).** Так как, хотя дружественная функция "знает" о закрытом поле класса, для которого она является дружественной, доступ к нему она может получить только через объект этого класса, который объявлен внутри функции и передан ей в качестве аргумента. ДФ вызывается для объекта класса.

**4. Какой вариант вызова ДФ правильный:**

a.show() или show(a)?

Если функция дружественная, то второй вариант.

**5. Наследуется ли дружественная функция?**

Дружественная функция не наследуется. То есть, если в базовый класс дружественная функция включается, то она не является таковой для производных классов.

**6. Может ли дружественная функция быть дружественной более чем к одному классу?**

Дружественная функция может быть дружественной более чем к одному классу.

**7. Для решения каких задачах есть смысл использовать функцию, дружественную двум и более классам?**

Такая дружественная функция полезна тогда, когда у двух и более разных классов имеется нечто общее, с чем необходимо произвести какие-то действия, например, сравнение или арифметические действия закрытых полей объектов, принадлежащих разным классам.

**8. Когда нужно использовать дружественные классы?**

С одной стороны, дружественные функции повышают гибкость языка, но, с другой стороны, они не соответствуют принципу ограничения доступа к данным, в соответствии с которым только методы класса могут иметь доступ к сокрытым данным класса. Множество дружественных функций фактически стирают границы между классами, поэтому они должны быть разбросаны по программе и встречаться как можно реже.

**9. Какому принципу ООП не соответствуют дружественные функции?**

Инкапсуляция.

**10. Как можно ограничить использование ДФ?**

Не использовать их, использовать открытые методы класса. Или каждый раз наследуя один класс от дружественного, в таком случае, ДФ не наследуется.

**11. В каких крайних случаях нужно использовать ДФ?**

Если требуется для базового класса сделать доступ к приватным данным, а для наследуемых нет. В этом случае играет роль то, что дружественные функции не наследуются.