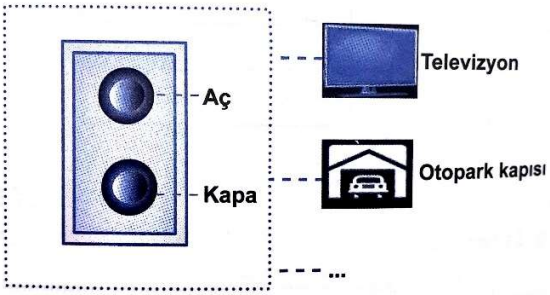
ÖDEV 5:

CEVAPLAR

1. Aşağıdaki şekilde de gösterildiği gibi, çok amaçlı bir uzaktan kumanda yapılmak istenmektedir. Kumanda üzerinde açma ve kapa işlemlerini gerçekleştiren iki adet düğme vardır. Bu düğmelerin, kumanda edeceği cihazlar (her cihaz mutlaka, ac() ve kapat() olarak iki fonksiyona sahiptir.) farklılaşabilmektedir. Aşağıda main fonksiyonu ve örnek çıktıda da görüldüğü gibi, bir UzaktanKumanda nesnesi yaratılmış ve nesnenin kumanda edeceği cihaz olarak televizyon belirtilmiştir. Böylece, kumanda aç ve kapat düğmeleri televizyon açma ve kapama işlevini yerine getirmektedir. Daha sonra aynı kumanda nesnesine otopark kapısı atanabilmektedir. Verilen main fonksiyonu ve örnek çıktıyı dikkate alarak;

a) UML sınıf diyagram çiziniz.
b) Main programın çalışması için gereken sınıf ve fonksiyon kodlarını yazınız.



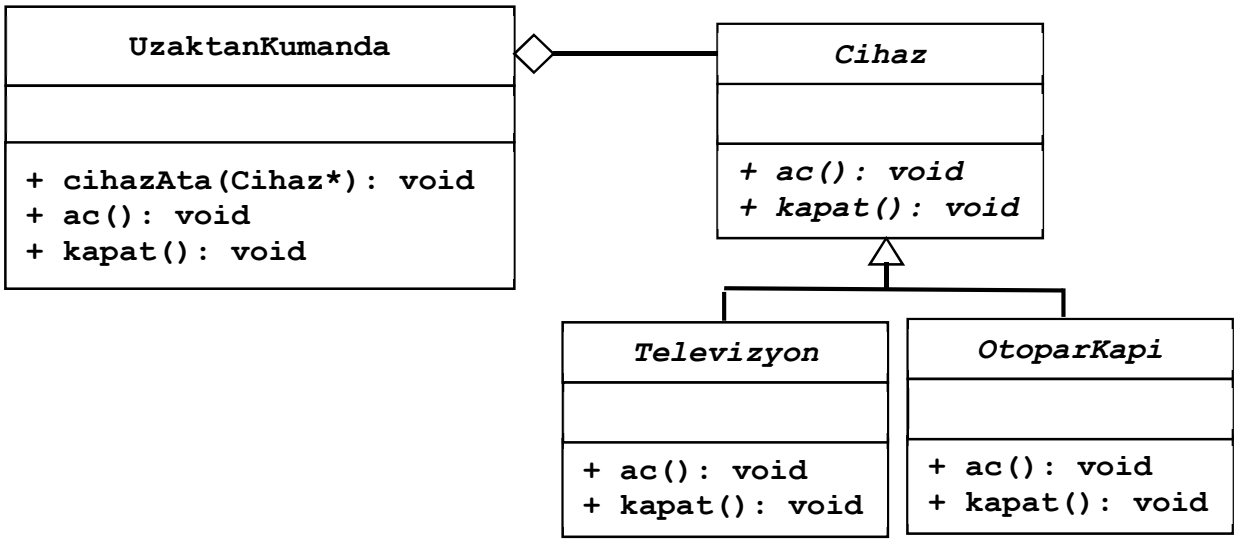

```
int main(){
     UzaktanKumanda kumanda;
     Televizyon tv;
     kumanda.cihazAta(&tv);
     kumanda.ac();
     kumanda.kapat();

     OtoparkKapi kapi;
     kumanda.cihazAta(&kapi);
     kumanda.ac();
     kumanda.kapat();

     return 0;
}
```

Örnek Çıktı:

```
Televizyon aciliyor...
Televizyon kapaniyor...
Otopark kapısı aciliyor...
Otopark kapısı kapaniyor...
```

Kaynak: Ç. Turhan ve F.C. Serçe, C++ Dersi: Nesne Tabanlı Programlama, 2nci Baskı, 2014.

Cevap: (a)



(b)

```
#include <iostream>
using namespace std;

class Cihaz {
public:
	virtual void ac() = 0;
	virtual void kapat() = 0;
};

class Televizyon:public Cihaz {
public:
	void ac() {
		cout << "Televizyon aciliyor..." << endl;
	}
	virtual void kapat() {
		cout << "Televizyon kapaniyor..." << endl;
	}
};

class OtoparkKapi :public Cihaz {
public:
	void ac() {
		cout << "Otopark kapisi aciliyor..." << endl;
	}
	virtual void kapat() {
		cout << "Otopark kapisi kapaniyor..." << endl;
	}
};
```

```
class UzaktanKumanda {
private:
	Cihaz *cihaz;
public:
	void cihazAta(Cihaz *c) {
		cihaz = c;
	}
	void ac() {
		cihaz->ac();
	}
	virtual void kapat() {
		cihaz->kapat();
	}
};

int main() {
	UzaktanKumanda kumanda;
	Televizyon tv;
	kumanda.cihazAta(&tv);
	kumanda.ac();
	kumanda.kapat();

	OtoparkKapi kapi;
	kumanda.cihazAta(&kapi);
	kumanda.ac();
	kumanda.kapat();

	return 0;
}
```

2. Bir üniversitede öğrenciler derslerde 0-100 arası not almaktadır. Öğrencilerin not ortalaması (gno) aşağıdaki formülle hesaplanır.

$$gno = \sum(not * kredi) / \sum kredi$$

Öğrencilerin not ortalamalarını bulmak amacıyla `Ogrenci` isimli aşağıdaki bileşenlerden oluşan bir sınıf tasarlayınız.

- Veri üyeleri: `ogrNo`, `gno`
- `hesap()` fonksiyonu: bir öğrencinin numarasını, kaç ders aldığını ve aldığı tüm derslerin adını, ders notunu ve kredisini okur ve öğrencinin not ortalamasını hesaplar. Herhangi bir not yanlış girilmişse throw ile hata fırlatılır ve örnek çıktıda görüldüğü gibi ekrana mesaj yazdırılır.
- `main()` fonksiyonu bir öğrencinin bilgilerini okur ve not ortalamasını ekrana yazdırır.

Örnek Çıktı:

```
Ogrencinin numarasini girin: 152120091010 ↵
Ogrencinin ders sayisini girin: 4 ↵
Derslerin isim not ve kredi bilgisini girin:
COMPE112 80 3 ↵
MATH276 812 4 ↵
```

```
Hata: MATH276 notu hatali olarak 812 girildi.
MATH276 82 4 ↵
COMPE223 90 3 ↵
IE220 70 3 ↵
Ogrenci No: 152120091010 Not Ortalamasi: 80.62
```

Cevap:

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

class Ogrenci {
public:
	Ogrenci() : ogrNo(-1), gno(0), dersSayisi(0)
			, notKredi(0), toplamKredi(0) {}

	void hesap() {
		string dersAdi;
		float dersNotu;
		float kredi;
		stringstream ss;
		if (ogrNo == -1) {
			cout << "Ogrencinin numarasini girin: ";
			cin >> ogrNo;
		}
		if (dersSayisi == 0) {
			cout << "Ogrencinin ders sayisini girin:";
			cin >> dersSayisi;
			cout << "Derslerin isim not ve kredi bilgisini girin:";
		}
		while (dersSayisi) {
			cin >> dersAdi >> dersNotu >> kredi;
			if ((dersNotu < 0) || (dersNotu>100)) {
				ss << "Hata: " << dersAdi
				   << " notu hatali olarak "
				   << dersNotu << "girildi.";
				throw ss.str();
			}
			else{
				notKredi+=(dersNotu*kredi);
				toplamKredi+=kredi;
			}
			dersSayisi--;
		}
		gno = notKredi / toplamKredi;
		cout << "Ogrenci No: "<<ogrNo<<" Not Ortalamasi: "<<gno;
	}
	int dersSayisiAl() {
		return dersSayisi;
	}
```

```
private:
	long ogrNo;
	float gno;
	int dersSayisi;
	float notKredi;
	float toplamKredi;
};

int main() {
	Ogrenci ogrenci;

	while (true) {
		try {
			ogrenci.hesap();
		}
		catch (string s) {
			cout << s << endl;
		}
		if (!ogrenci.dersSayisiAl())
			break;
	}
	return 0;
}
```

3. Aşağıda verilen `Kare` sınıfının kenar veri üyesi tamsayı tipinde tanımlanmıştır. `kenar`'ın herhangi bir veri tipinde olabilmesi için `Kare` sınıfını şablon olarak tekrar yazın ve kenarı 2.5 ve 9 olan iki `Kare` nesnesinin alanlarını ekrana yazdıran bir `main()` fonksiyonu yazınız.

```
class Kare{
	int kenar;
public:
	void kenarAta(int k){
		kenar=k;
	}
	int kenarAl(){
		return kenar;
	}
	int alan();
};

int Kare::alan()
{
	return kenar*kenar;
}
```

Cevap:

```
#include <iostream>
using namespace std;

template<typename T>
class Kare {
	T kenar;
public:
	void kenarAta(T k) {
		kenar = k;
	}
	T kenarAl() {
		return kenar;
	}
	T alan();
};

template<typename T>
T Kare<T>::alan()
{
	return kenar*kenar;
}


int main(){
	Kare<int> intKare;
	intKare.kenarAta(9);

	cout << "Kenari " << intKare.kenarAl()
		<< " olan karenin alani: " << intKare.alan() << endl;

	Kare<float> floatKare;
	floatKare.kenarAta(2.5);

	cout << "Kenari " << floatKare.kenarAl()
		<< " olan karenin alani: " << floatKare.alan() << endl;

	return 0;
}
```