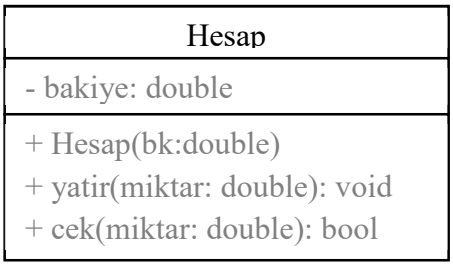
152113017: NESNE TABANLI PROGRAMLAMA I
FINAL EXAM (26.01.2022)
Kitap ve notlar kapalı
Toplam: 100 puan, 3 Soru, Süre: 75 dakika

S.1	S.2	S.3	Toplam

Adı-Soyadı: CEVAPLAR	Numara: CEVAPLAR

S.1 (35p.) Verilen UML diyagrama göre aşağıdaki soruları cevaplayınız. constructor fonksiyonu varsayılan olarak bakiyeyi sıfırlar, bir değer (bk) verilmiş ise değeri bakiyeye atar. yatir(miktar) fonksiyonu miktari hesap bakiyesine ekler. cek(miktar) fonksiyonu, miktari bakiyeden düşer. Bakiye yeterli ise, true değilse false döndürür.



(a) (15p) Hesap sınıfını yazınız.

```
class Hesap {
      double bakiye;
public:
      Hesap(double bk);
      void yatir(double miktar);
      bool cek(double miktar);
};

Hesap::Hesap(double bk = 0) {
      if (bk < 0)
            bakiye = 0;
      else
            bakiye = bk;
}

void Hesap::yatir(double miktar)
{
      if (miktar > 0)
            bakiye += miktar;
}

bool Hesap::cek(double miktar)
{
      if (miktar > bakiye)
            return false;
      else {
            bakiye -= miktar;
            return true;
      }

}
```

(b) (10p) Bir operatör fonksiyonunu Hesap sınıfının üyesi olarak yazınız. Bu operatör ile şu işlem yapılabilecektir:

```
Hesap hesap(1000);
hesap+=500; // Bu satır işletilince,
            // bakiye 1500 olur.
```

```
Hesap operator+=(double miktar) {
      bakiye += miktar;
}
```

(c) (10p) Hesap sınıfında istisnai durum yönetimi (exception handling) yapalım. Constructor fonksiyonunda, argüman (bk) olarak negatif değer girilirse, "geçersizBakiye" olarak string fırlatılacaktır (throw). Main fonsiyonda, fırlatılan değer yakalanıp, ekrana yazdırılacaktır (Bir Hesap nesnesi yaratılmasında, try ve catch kullanımı). **(Not: sadece tüm Hesap sınıfını yazmayın. Sadece istenen eklentiyi gösterin. (a) şıkkı üzerinde de gösterebilirsiniz.)**

```
Hesap::Hesap(double bk = 0) {
	if (bk < 0)
		throw (string)"gecersizBakiye";
	else
		bakiye = bk;
}
```

```
int main() {

	try {
		Hesap hesap(-1);
	}
	catch (string str) {
		cout << str << endl;
	}
	return 0;
}
```

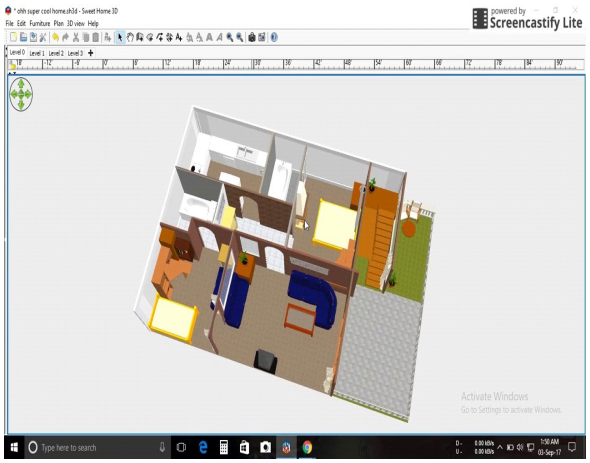
S.2 (20p.) Aşağıda bir Table sınıfı verilmiştir. Bu sınıfa ait iki üyenin (key ve data) istenen tipte belirlenebilmesi istenmektedir. Örneğin, key üyesi char, string, vb. tipte seçilebilirken, data üyesi de int, double, string, vb. seçilebilmelidir. Yani bu iki üye birbirinden farklı ve istenen veri tipinde olabilmelidir. Buna göre sınıf kodlarını ve uygulama programının (main fonksiyon) güncellenmiş halini yazınız.

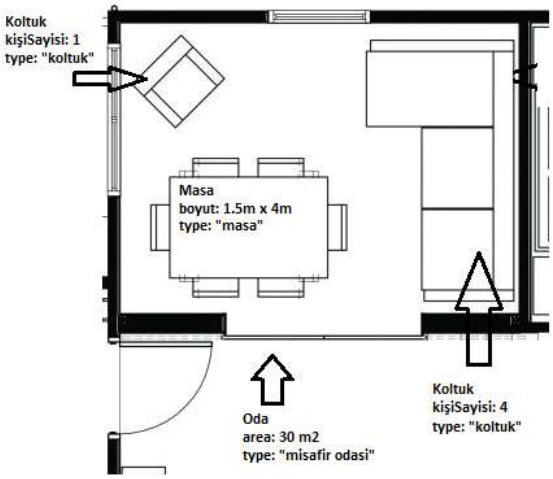
```
class Table {
	int key;
	string data;
public:
	Table(int _key, string _data)
		:key(_key), data(_data) {}
	bool getData(int _key
		, string &_data) const;
};

bool Table::getData(int _key
		, string &_data) const
{
	if (key == _key) {
		_data = data;
		return true;
	}
	else{
		return false;
	}
}

int main() {
	int k=1;
	string data="test";

	Table table(k, data);
	return 0;
}
```

Cevap:

```
template <typename P_key,typename P_data>
class Table {
	P_key key;
	P_data data;
public:
	Table(P_key _key, P_data _data)
		:key(_key), data(_data) {}
	bool getData(P_key _key
		, P_data &_data) const;
};

template <typename P_key, typename P_data>
bool Table<P_key, P_data>::getData(P_key _key
			, P_data &_data) const
{
	if (key == _key) {
		_data = data;
		return true;
	}
	else{
		return false;
	}
}

int main() {
	int k=1;
	string data="test";

	Table<int,string> table(1, "test");

	return 0;
}
```

S.3 (45p.) Ev Tasarım Yazılımı: Ev tasarımlarında, ev içindeki eşya yerleşimini göstermek için yazılımlar kullanılmaktadır. Bu yazılımlarda, yandaki resimde de görüldüğü gibi 2B ve 3B olarak bina kat planı üzerinde eşyaların yerleşimi yapılmaktadır.

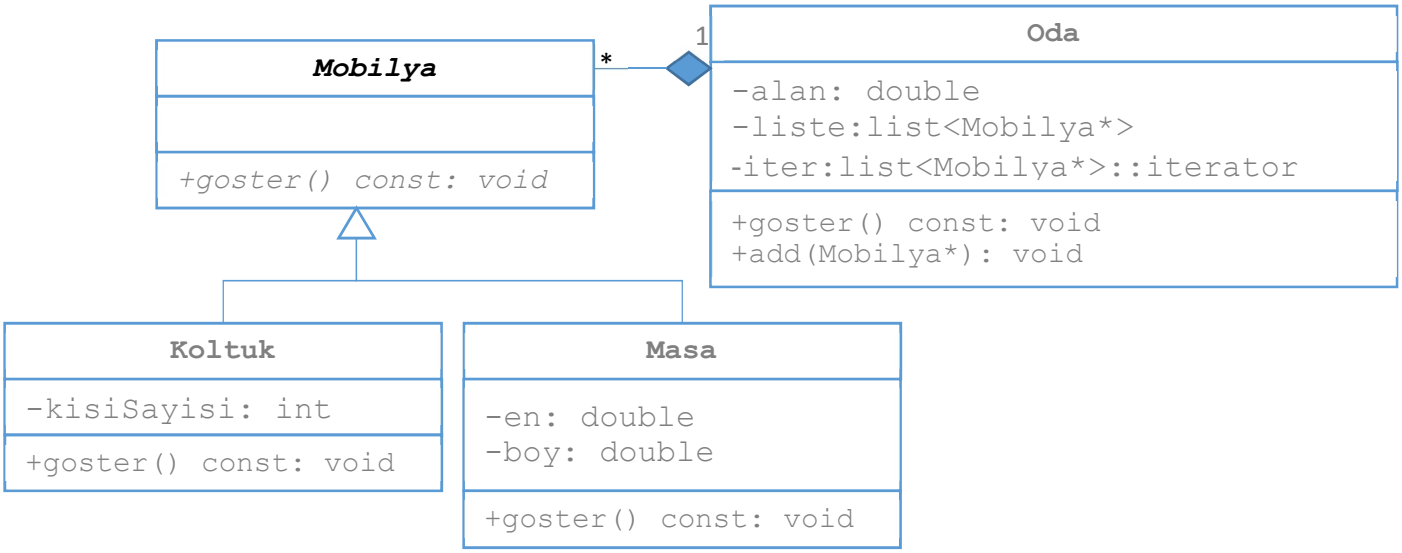




Sizden bir nesne tabanlı yazılım tasarımı istenmektedir. Bu tasarımda iki tip `mobilya` bulunmaktadır. Bunlar `Koltuk` ve `Masa`. `Koltuk`, `kişi sayısı` bilgisi içerirken, `Masa` ise `en` ve `boy` bilgisi içermektedir. Her `mobilya` için mutlaka özellik gösterimi (`goster`) mümkündür. Ayrıca, `oda` vardır. `Oda`, `alan` bilgisi ve `oda`da bulunan mobilyaların listesini (`std::list`) içermektedir. `Oda`daki `liste`ye `masa`lar ve `koltuk`lar eklenebilmektedir (`add`). `Oda`nın `liste`sindeki tüm `mobilya`ların özellikleri listelenerek `göster`ilebilmektedir. Bir oda yok edildiğinde, odadaki mobilyalar da yok edilmektedir.

(Tasarımınızdaki isimlendirmelerde, öncelikle problem açıklamasında farklı fontta verilen kelime ve türevlerini kullanmanız beklenmektedir.)

(a) (15p) Tasarımınızın UML diyagramını çiziniz. (Yukarıdaki açıklamalara uygun olarak, polymorphic yapı, abstract (soyut) sınıf, pure virtual (saf sanal) fonksiyon, std::list, veri saklama mutlaka kullanılmalı ve tasarımda bunlar görülmelidir. Diyagramda, sınıflar arası ilişkiler doğru çizimlerle gösterilmelidir. ilişkinin composition, aggregation ve inheritance hangisi olduğu anlaşılabilmelidir. İlişkilerde, çarpanlar (multiplity) gösterimelidir.)



(b) (20p) Sadece sınıfları kodlayınız. (Yukarıdaki açıklamalara uygun olarak, polymorphic yapı, abstract (soyut) sınıf, pure virtual (saf sanal) fonksiyon, std::list, veri saklama mutlaka kullanılmalı ve tasarımda bunlar görülmelidir.)

```
class Mobilya {
public:
      Mobilya() {}
      virtual void goster() const=0;
};

class Koltuk :public Mobilya {
      int kisiSayisi;
public:
      Koltuk(int kisi):kisiSayisi(kisi){}
      void goster() const;
};

class Masa :public Mobilya {
      double en;
      double boy;
public:
      Masa(double e=0, double b=0)
          :en(e), boy(b){}
      void goster() const;
};
class Oda {
      double alan;
      list<Mobilya*> liste;
      list<Mobilya*>::iterator iter;
public:
      Oda(double Alan=0):alan(Alan){}
      void add(Mobilya*);
      void goster();
      ~Oda();
};
```

```
void Koltuk::goster() const
{
      cout << "Kisi Sayisi: "
           << kisiSayisi << endl;
}


void Masa::goster() const
{
      cout << "En: " << en
           <<", Boy: "<<boy<< endl;
}


void Oda::add(Mobilya* m)
{
      liste.push_back(m);

}

void Oda::goster()
{
      cout << "Oda Alani: "
           << alan << endl;
      for(iter = liste.begin(); iter !=
liste.end(); iter++)
            (*iter)->goster();

}

Oda::~Oda()
{
      for (iter = liste.begin(); iter !=
liste.end(); iter++)
            delete (*iter);
}
```

(c) (10p) Yukarıdaki programda, pure virtual function (saf sanal fonksiyon) hangisidir? Abstract (soyut) sınıf hangisidir? Neden pure virtual fonksiyona ihtiyaç duyulmaktadır? Abstract sınıfın özelliği ve tasarımdaki fonksiyonları nedir? Açıklayınız.

pure virtual function: `virtual void goster() const=0;`
abstract sınıf: `class Mobilya`
Pure virtual fonksiyon, belli bir tasarıma zorlamak için kullanılır. Bu fonksiyona sahip soyut sınıftan miras yolu ile türetilen somut sınıflar, kodlanırken saf sanal fonksiyonu kodlamak zorunda kalırlar.
Abstract (soyut) sınıf, en az bir adet saf sanal fonksiyona sahip olan sınıftır. Soyut sınıftan nesne yaratılamaz. Tasarımda, bu sınıftan miras yolu ile türetilen sınıflar için arayüz görevini üstlenir.

Başarılar…