C++期末考试 2015/06/26 (8:10-9:40)

事件	时间	
提交开始	2016-06-08	0:0
提交结束	2017-06-01	10:0

1-2 of 2

学号	
姓名	
班级	

保存

第1题，class *Song*

Please write the following functions in class *Song*.

- new Song::Song(string name, Artist & artist, Album & album)

This problem requires around 5 lines of code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testSong --
song#1: UNKNOWN ARTIST, UNKNOWN ALBUM
song#2: UNKNOWN ARTIST, UNKNOWN ALBUM
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

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

class Artist // 音乐艺术家(歌唱者)
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album // 音乐专辑
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

/////////////////////////
// Song
/////////////////////////

class Song
{
public:
    string name;
    Artist * artist;
    Album * album;

public:
    Song(string name, Artist & artist, Album & album); // TODO

    void print() {
        cout << name << ": " << artist->getName() << ", " << album->getName() << endl;
    }

};

#include "source"

/////////////////////////
// Test
/////////////////////////
```

```
void testSong() {
    cout << "-- testSong --" << endl;
    Artist artist;
    Album album;
    Song song1("song#1", artist, album);
    Song song2("song#2", artist, album);
    song1.print();
    song2.print();
}

int main() {
    testSong();
}
```

答案程序

```
Song::Song(string name, Artist & artist, Album & album) {
    this->name = name;
    this->artist = &artist;
    this->album = &album;
}
```

文本编辑框

保存和测试对错 答案正确

第2题, class *Song*

注意：本题需在 第1题 的基础上完成。

Please write the following functions in class *Song*.

- Song::Song(string name, Artist & artist, Album & album)
- new string Song::getName()
- new Artist & Song::getArtist()
- new Album & Song::getAlbum()

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 15 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testSong --
song#1: UNKNOWN ARTIST, UNKNOWN ALBUM
song#2: UNKNOWN ARTIST, UNKNOWN ALBUM
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

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

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

////////////////////////////
// Song
////////////////////////////

class Song
{
public:
    string name;
    Artist * artist;
    Album * album;

public:
    Song(string name, Artist & artist, Album & album);

    string getName(); // TODO

    Artist & getArtist(); // TODO

    Album & getAlbum(); // TODO

};

#include "source"

void print(Song & song) {
    cout << song.getName() << ": " << song.getArtist().getName() << ", " << song.getAlbum().getName() << endl;
}

/////////////////////////
// Test
/////////////////////////

void testSong() {
    cout << "-- testSong --" << endl;
    Artist artist;
    Album album;
    Song song1("song#1", artist, album);
    Song song2("song#2", artist, album);
    print(song1);
    print(song2);
}

int main() {
    testSong();
}
```

答案程序

```
Song::Song(string name, Artist & artist, Album & album) {
    this->name = name;
    this->artist = &artist;
    this->album = &album;
}

string Song::getName() { // TODO
    return this->name;
}

Artist & Song::getArtist() { // TODO
    return *(this->artist);
}

Album & Song::getAlbum() { // TODO
    return *(this->album);
}
```

文本编辑框

保存和测试对错 答案正确

第3题, class *Song*

注意：本题需在 第1-2题 的基础上完成。

Please write the following functions for class *Song*.

- Song::Song(string name, Artist & artist, Album & album)
- string Song::getName()
- Artist & Song::getArtist()
- Album & Song::getAlbum()
- new ostream & operator << (ostream & out, Song & song)

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 5 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testSong --
song#1: UNKNOWN ARTIST, UNKNOWN ALBUM
song#2: UNKNOWN ARTIST, UNKNOWN ALBUM
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

/////////////////////////
// Song
/////////////////////////

class Song
{
public:
    string name;
    Artist * artist;
    Album * album;

public:
    Song(string name, Artist & artist, Album & album);

    string getName();

    Artist & getArtist();

    Album & getAlbum();

};

#include "source"

/////////////////////////
// Test
/////////////////////////

void testSong() {
    cout << "-- testSong --" << endl;
    Artist artist;
    Album album;
    Song song1("song#1", artist, album);
    Song song2("song#2", artist, album);
    cout << song1 << endl; // TODO
    cout << song2 << endl;
}

int main() {
    testSong();
}
```

答案程序

```cpp
Song::Song(string name, Artist & artist, Album & album) {
    this->name = name;
    this->artist = &artist;
    this->album = &album;
}

string Song::getName() { // TODO
    return this->name;
}

Artist & Song::getArtist() { // TODO
    return *(this->artist);
}

Album & Song::getAlbum() { // TODO
    return *(this->album);
}

ostream & operator << (ostream & out, Song & song) {
    cout << song.getName() << ": " << song.getArtist().getName() << ", " << song.getAlbum().getName() << endl;
    return out;
}
```

文本编辑框

保存和测试对错

第4题, class *Artist (音乐艺术家)*

Please write the following functions for class *ArtistImpl*.

- new void ArtistImpl::addSong(Song & song)

This problem requires around 15 lines of code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testArtist --
artist#1: song#1
artist#1: song#1, song#2
caught: SongExistedException
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

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

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

class Song
{
public:
    string name;

public:
    Song(string name) {
        this->name = name;
    }

    string getName() {
        return this->name;
    }

};

/////////////////////////
// Artist
/////////////////////////

```

```
class SongExistedException {};

class ArtistImpl : public Artist
{
private:
    string name;
    vector<Song> songs;

public:
    ArtistImpl(string name) {
        this->name = name;
    }

    virtual string getName() {
        return this->name;
    }

public:

    void print() {
        cout << this->getName() << ": ";
        for (int i = 0; i < this->songs.size(); ++ i) {
            cout << (i == 0 ? "" : ", ") << this->songs[i].getName();
        }
        cout << endl;
    }

    void addSong(Song & song); // TODO, may throw SongExistedException
};

#include "source"

/////////////////////////
// Test
/////////////////////////

void testArtist() {
    cout << "-- testArtist --" << endl;
    ArtistImpl artist1("artist#1");
    Song song1("song#1");
    Song song2("song#2");

    try {
        artist1.addSong(song1);
        artist1.print();
        artist1.addSong(song2);
        artist1.print();
        artist1.addSong(song2);
        artist1.print();
    } catch (SongExistedException & ex) {
        cout << "caught: SongExistedException" << endl;
    }

}

int main() {
    testArtist();
}
```

答案程序

```
void ArtistImpl::addSong(Song & song) {
    // if song is in songs, throw an exception
    for (int i = 0; i < songs.size(); ++ i) {
        if (songs[i].getName() == song.getName()) {
            throw SongExistedException(); // return
        }
    }
    // else add song into songs
    songs.push_back(song);
}
```

文本编辑框

保存和测试对错 答案正确

第5题, class *Artist* (音乐艺术家)

注意: 本题需在 第4题 的基础上完成。

Please write the following functions for class *ArtistImpl*.

- void ArtistImpl::addSong(Song & song)
- new Song ArtistImpl::removeSong(Song & song)

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 15 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testArtist --
artist#1: song#1, song#2
artist#1: song#1
caught: SongNotFoundException
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

class Song
{
public:
    string name;

public:
    Song(string name) {
        this->name = name;
    }

    string getName() {
        return this->name;
    }

};

//////////////////////////
// Artist
//////////////////////////

class SongExistedException {};

class SongNotFoundException {};

class ArtistImpl : public Artist
{
private:
    string name;
    vector<Song> songs;

public:
    ArtistImpl(string name) {
        this->name = name;
    }

    virtual string getName() {
        return this->name;
    }

public:

    void print() {
        cout << getName() << ": ";
        for (int i = 0; i < songs.size(); ++ i) {
            cout << (i == 0 ? "" : ", ") << songs[i].getName();
        }
        cout << endl;
    }

    void addSong(Song & song); // may throw SongExistedException

    Song removeSong(Song & song); // TODO, return the removed song, may throw SongNotFoundException
};

#include "source"

/////////////////////////
// Test
/////////////////////////

void testArtist() {
    cout << "-- testArtist --" << endl;
    ArtistImpl artist1("artist#1");
    Song song1("song#1");
    Song song2("song#2");

    try {
        artist1.addSong(song1);
        artist1.addSong(song2);
        artist1.print();
        artist1.removeSong(song2);
        artist1.print();
        artist1.removeSong(song2);
        artist1.print();
    } catch (SongNotFoundException & ex) {
        cout << "caught: SongNotFoundException" << endl;
    }

}

int main() {
    testArtist();
}
```

答案程序

```
void ArtistImpl::addSong(Song & song) {
    // if song is in songs, throw an exception
    for (int i = 0; i < songs.size(); ++ i) {
        if (songs[i].getName() == song.getName()) {
            throw SongExistedException(); // return
        }
    }
    // else add song into songs
    songs.push_back(song);
}

Song ArtistImpl::removeSong(Song & song) {
    // if song not exists, throw exception
    int index = -1;
    for (int i = 0; i < songs.size(); ++ i) {
        if (songs[i].getName() == song.getName()) {
            index = i;
            break;
        }
    }
    if (index == -1) throw SongNotFoundException();
    // else remove song
    song = songs[index];
    for (int i = index + 1; i < songs.size(); ++ i) {
        songs[i - 1] = songs[i];
    }
    songs.pop_back();
    return song;
}
```

文本编辑框

保存和测试对错 答案正确

第6题，class *Artist (音乐艺术家)*

注意：本题需在 第4-5题 的基础上完成。

Please write the following functions for class *ArtistImpl*.

- void ArtistImpl::addSong(Song & song)
- Song ArtistImpl::removeSong(Song & song)
- new int ArtistImpl::numOfSongs()
- new Song & ArtistImpl::getSong(int index)
- new ostream & operator << (ostream & out, ArtistImpl & artist)

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 20 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testArtist --
artist#1: song#1
artist#1: song#1, song#2
artist#1: song#2
caught: SongNotFoundException
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

```cpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

class Song
{
public:
    string name;

public:
    Song(string name) {
        this->name = name;
    }

    string getName() {
        return this->name;
    }

};

/////////////////////////
// Artist
/////////////////////////

class SongExistedException {};

class SongNotFoundException {};

class ArtistImpl : public Artist
{
private:
    string name;
    vector<Song> songs;

public:
    ArtistImpl(string name) {
        this->name = name;
    }

    virtual string getName() {
        return this->name;
    }

public:

    void addSong(Song & song); // may throw SongExistedException

    Song removeSong(Song & song); // return the removed song, may throw SongNotFoundException

    Song & getSong(int index); // TODO, may throw SongNotFoundException

    int numOfSongs(); // TODO
};

#include "source"

//////////////////////////
// Test
//////////////////////////

void testArtist() {
    cout << "-- testArtist --" << endl;
    ArtistImpl artist1("artist#1");
    Song song1("song#1");
    Song song2("song#2");

    try {
        artist1.addSong(song1);
        cout << artist1 << endl; // TODO
        artist1.addSong(song2);
        cout << artist1 << endl;
        artist1.removeSong(song1);
        cout << artist1 << endl;
        artist1.getSong(artist1.numOfSongs());
    } catch (SongNotFoundException & ex) {
        cout << "caught: SongNotFoundException" << endl;
    }
}

int main() {
    testArtist();
}
```

答案程序

1

文本编辑框

保存和测试对错

第7题, class *Album* (音乐专辑)

注意: 本题需在 第1-6题 的基础上完成。

Please write the following functions for class *AlbumImpl*.

- new AlbumImpl::AlbumImpl(string albumName)

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 5 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testAlbum --
album#1
Songs (0)
```

注意: 请注意程序风格 (共5点), 将检查并扣分。

主程序

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

```

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

//////////////////////////
// Song
//////////////////////////

class Song; // TODO: need to include your complete class Song later.

//////////////////////////
// Album
//////////////////////////

class AlbumImpl : public Album
{
private:

    /*
    Hint: We will use class ArtistImpl to store songs
    */
    Artist * album; // it is used to store all songs & the name of the album
    vector<Artist *> artists; // it is used store artists

public:

    virtual string getName();

    int numOfSongs(); // return the number of songs

    ~AlbumImpl() {
        delete this->album;
        for (int i = 0; i < artists.size(); ++ i) {
            delete this->artists[i];
        }
    }

    AlbumImpl(string albumName); // TODO

};

/*
TODO: Please also include
- class Song
- class ArtistImpl
*/
#include "source"

string AlbumImpl::getName() {
    ArtistImpl * album = (ArtistImpl *)this->album;
    return album->getName();
}

int AlbumImpl::numOfSongs() { // return the number of songs
    ArtistImpl * album = (ArtistImpl *)this->album;
    return album->numOfSongs();
}

void print(ostream & out, AlbumImpl & album) {
    cout << album.getName() << endl;
    cout << "Songs (" << album.numOfSongs() << ")" << endl;
}

//////////////////////////
// Test
//////////////////////////

void testAlbum() {
    cout << "-- testAlbum --" << endl;
    AlbumImpl album1("album#1");
    print(cout, album1);

}

int main() {
    testAlbum();
}
```

答案程序

1

文本编辑框

保存和测试对错

第8题, class *Album* (音乐专辑)

注意: 本题需在 第1-7题 的基础上完成。

Please write the following functions for class *AlbumImpl*.

- AlbumImpl::AlbumImpl(string albumName)
- new Song & AlbumImpl::operator [] (int songIndex)
- new void AlbumImpl::addSong(string songName, string artistName)

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 30 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testAlbum --
album#1
Songs (1):
  song#1: artist#1, album#1
Artists (1):
  artist#1

album#1
Songs (2):
  song#1: artist#1, album#1
  song#2: artist#1, album#1
Artists (1):
  artist#1

album#1
Songs (3):
  song#1: artist#1, album#1
  song#2: artist#1, album#1
  song#3: artist#2, album#1
Artists (2):
  artist#1
  artist#2

caught: SongExistedException
```

注意: 请注意程序风格 (共5点), 将检查并扣分。

主程序

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

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};
```

```cpp

/////////////////////////
// Song
/////////////////////////

class Song;

/////////////////////////
// Album
/////////////////////////

class AlbumImpl : public Album
{
private:

    Artist * album; // use to store all songs & the name of the album
    vector<Artist *> artists;

public:

    virtual string getName();

    int numOfSongs(); // return the number of songs


    ~AlbumImpl() {
        delete this->album;
        for (int i = 0; i < artists.size(); ++ i) {
            delete this->artists[i];
        }
    }

    AlbumImpl(string albumName);

    Song & operator [] (int songIndex); // TODO, may throw SongNotFoundException

    void addSong(string songName, string artistName); // TODO, may throw SongExistedException

    void print(); // NEW

};

/*
Please also include
- class Song
- class ArtistImpl
*/
#include "source"

string AlbumImpl::getName() {
    ArtistImpl * album = (ArtistImpl *)this->album;
    return album->getName();
}

int AlbumImpl::numOfSongs() { // return the number of songs
    ArtistImpl * album = (ArtistImpl *)this->album;
    return album->numOfSongs();
}

void AlbumImpl::print() {
    cout << this->getName() << endl;
    cout << "Songs (" << this->numOfSongs() << "):" << endl;
    for (int i = 0; i < this->numOfSongs(); ++ i) {
        cout << "  " << this->operator[](i) << endl;
    }
    cout << "Artists (" << this->artists.size() << "):" << endl;
    for (int i = 0; i < this->artists.size(); ++ i) {
        ArtistImpl * artist = (ArtistImpl *)(this->artists[i]);
        cout << "  " << artist->getName() << endl;
    }
    cout << endl;
}

/////////////////////////
// Test
/////////////////////////

void testAlbum() {
    cout << "-- testAlbum --" << endl;
    AlbumImpl album1("album#1");

    try {
        album1.addSong("song#1", "artist#1");
        album1.print();
        album1.addSong("song#2", "artist#1");
        album1.print();
        album1.addSong("song#3", "artist#2");
        album1.print();
        album1.addSong("song#2", "artist#1");
        album1.print();
    } catch (SongExistedException & ex) {
        cout << "caught: SongExistedException" << endl;
    }

}

int main() {
    testAlbum();
}
```

答案程序

```
int main() {
    testAlbum();
}
```

文本编辑框

保存和测试对错

第9题, class *Album* (音乐专辑)

注意：本题需在 第1-8题 的基础上完成。

Please write the following functions for class *AlbumImpl*.

- AlbumImpl::AlbumImpl(string albumName)
- Song & AlbumImpl::operator [] (int songIndex)
- void AlbumImpl::addSong(string songName, string artistName)
- new void AlbumImpl::removeSong(string songName)
- new ostream & operator << (ostream & out, AlbumImpl & album)

Note that you might want to copy some previous code to solve this problem.
By using previous code, this problem requires around 30 lines of new code.

EXAMPLE INPUT

无

EXAMPLE OUTPUT

```
-- testAlbum --
album#1
Songs (3):
  song#1: artist#1, album#1
  song#2: artist#1, album#1
  song#3: artist#2, album#1
Artists (2):
  artist#1: song#1, song#2
  artist#2: song#3

album#1
Songs (2):
  song#2: artist#1, album#1
  song#3: artist#2, album#1
Artists (2):
  artist#1: song#2
  artist#2: song#3

caught: SongNotFoundException
```

注意：请注意程序风格（共5点），将检查并扣分。

主程序

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

class Artist
{
public:
    virtual string getName() {
        return "UNKNOWN ARTIST";
    }
```

```
};

class Album
{
public:
    virtual string getName() {
        return "UNKNOWN ALBUM";
    }
};

//////////////////////////
// Song
//////////////////////////

class Song;

//////////////////////////
// Album
//////////////////////////

class AlbumImpl : public Album
{
private:

    Artist * album; // use to store all songs & the name of the album
    vector<Artist *> artists;

public:

    virtual string getName();

    int numOfSongs(); // return the number of songs


    ~AlbumImpl() {
        delete this->album;
        for (int i = 0; i < artists.size(); ++ i) {
            delete this->artists[i];
        }
    }

    AlbumImpl(string albumName);

    Song & operator [] (int songIndex); // may throw SongNotFoundException

    void addSong(string songName, string artistName); // may throw SongExistedException

    void removeSong(string songName); // TODO, may throw SongNotFoundException

};

/*
Please also include
- class Song
- class ArtistImpl
*/
#include "source"

string AlbumImpl::getName() {
    ArtistImpl * album = (ArtistImpl *)this->album;
    return album->getName();
}

int AlbumImpl::numOfSongs() { // return the number of songs
    ArtistImpl * album = (ArtistImpl *)this->album;
    return album->numOfSongs();
}

//////////////////////////
// Test
//////////////////////////

void testAlbum() {
    cout << "-- testAlbum --" << endl;
    AlbumImpl album1("album#1");

    try {
        album1.addSong("song#1", "artist#1");
        album1.addSong("song#2", "artist#1");
        album1.addSong("song#3", "artist#2");
        cout << album1 << endl; // TODO
        album1.removeSong("song#1");
        cout << album1 << endl;
        album1.removeSong("song#1");
    } catch (SongNotFoundException & ex) {
        cout << "caught: SongNotFoundException" << endl;
    }

}

int main() {
    testAlbum();
}
```

答案程序

```
asdfjas;dfjas
fasdfasdf
a
```

文本编辑框

保存和测试对错

************ 测试 ************

请使用以下快捷键盘操作在程序框之间拷贝程序

- Control + c **复制**
- Control + v **粘贴**

主程序

```
#include "source"
```

答案程序

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

int main() {
    int age;
    double speed;
    cin >> age;
    cin >> speed;
    cout << age << ", " << speed;
}
```

文本编辑框

测试数据

保存和测试

测试输出

保存于 9:6.55