Пример программы, реализующей сюжет сказки о курочке рябе
Обратите внимание, что сюжет реализован на основе паттерна «Цепочки ответственности».
Согласно методичке следует использовать другие поведенческие паттерны. А именно «Команда» или «Наблюдатель». Требуемый паттерн указан в таблице заданий.
Также в примере не реализована проверка конфигурации. Это следует сделать самостоятельно.

```
#include <iostream>
#include <stdio.h>
#include <fstream>
#include <cstdlib>
using namespace std;
class FairyPerson{
     protected:
       FairyPerson* next;
     public:
    FairyPerson() {this->next=0;}
     virtual void info()=0;
     virtual void analyzeCom(int com)=0;
     void setNext(FairyPerson* next) {
          this->next=next;
     }
};
class GrandMother;
class GrandFather:public FairyPerson {
     GrandMother* babushka;
     public:
     GrandFather():FairyPerson() {

     }
     void setGrandMother(GrandMother* babushka) {
          this->babushka=babushka;
     }
     GrandMother* getGrandMother() {
          return babushka;
     }
     void info() {
          cout << "дедушка: Я дедушка." << endl;
     }
     void crashEgg() {
          cout << "дедушка: бью яйцо!" << endl;
     }
     void cry() {
          cout << "дедушка: плачу." << endl;
     }
     void analyzeCom(int com) {
          if (com==1) crashEgg();
          else
          if (com==2) cry();
         else
          if (next!=0)
               next->analyzeCom(com);
     }
};
class GrandMother:public FairyPerson {
     GrandFather* dedushka;
     public:
     GrandMother():FairyPerson() {
```

```
	}
	void setGrandFather(GrandFather* dedushka) {
		this->dedushka=dedushka;
	}
	GrandFather* getGrandFather() {
		return dedushka;
	}
	void info() {
		cout << "Бабушка: Я бабушка" << endl;
	}
	void crashEgg() {
		cout << "Бабушка: бью яйцо!" << endl;
	}
	void cry() {
		cout << "Бабушка: плачу" << endl;
	}
	void analyzeCom(int com) {
		if (com==3) crashEgg();
		else
		if (com==4) cry();
	    else
		if (next!=0)
			next->analyzeCom(com);
	}
};

class Hen:public FairyPerson {
	public:
	Hen():FairyPerson() {
	}
	void info() {
		cout << "Курочка: Я курочка." << endl;
	}
	void createEgg() {
		cout << "Курочка: несу яйцо!" << endl;
	}
	void calm() {
		cout << "Курочка: успокаиваю дедушку и бабушку." << endl;
	}
	void analyzeCom(int com) {
		if (com==5) createEgg();
		else
		if (com==6) calm();
	    else
		if (next!=0)
			next->analyzeCom(com);
	}
};
class Mouse:public FairyPerson {
	public:
	Mouse():FairyPerson() {
	}
	void info() {
		cout << "Мышка: Я мышка." << endl;
	}
	void crashEgg() {
		cout << "Мышка: разбиваю яйцо!" << endl;
	}
```

```
     void analyzeCom(int com) {
          if (com==7) crashEgg();
          else
          if (next!=0)
               next->analyzeCom(com);
     }
};

class FactoryFairyPerson{
     public:
     virtual FairyPerson* createFairyPerson()=0;
};

class FactoryGrandFather:public FactoryFairyPerson{
   public:
     FairyPerson* createFairyPerson() {
       GrandFather* gf=new GrandFather();
       return gf;
     }
};
class FactoryGrandMother:public FactoryFairyPerson{
   public:
     FairyPerson* createFairyPerson() {
       GrandMother* gm=new GrandMother();
       return gm;
     }
};
class FactoryHen:public FactoryFairyPerson{
   public:
     FairyPerson* createFairyPerson() {
       Hen* hen=new Hen();
       return hen;
     }
};
class FactoryMouse:public FactoryFairyPerson{
   public:
     FairyPerson* createFairyPerson() {
       Mouse* m=new Mouse();
       return m;
     }
};
int main() {
     FactoryGrandFather* fgf = new FactoryGrandFather();
     FactoryGrandMother* fgm = new FactoryGrandMother();
     FactoryHen* fhen        = new FactoryHen();
     FactoryMouse* fmouse    = new FactoryMouse();
     GrandFather* gf = (GrandFather*) fgf    -> createFairyPerson();
     GrandMother* gm = (GrandMother*) fgm    -> createFairyPerson();
     Hen* hen        = (Hen*)         fhen   -> createFairyPerson();
     Mouse* mouse    = (Mouse*)       fmouse -> createFairyPerson();
     gf -> setGrandMother(gm);
     gm -> setGrandFather(gf);
     cout << "*****************************************"<< endl;
     cout << "* Курсовой проект по дисциплине             *"<< endl;
     cout << "* Объектно-ориентированное программирование *"<< endl;
     cout << "* Тема: Сказка 'Курочка ряба'               *"<< endl;
     cout << "* выполнил студент группы ИТ-961            *"<< endl;
     cout << "* Петров Петр Петрович                      *"<< endl;
```

```
      cout << "*******************************************"<< endl;
      gf -> getGrandMother()->info();
      gf->getGrandMother()->info();
      gm  -> getGrandFather()->info();
      gf  -> setNext(gm);
      gm  -> setNext(hen);
      hen -> setNext(mouse);
     int cmd;
      char buff[128];
      ifstream fin("cmd.txt");
      if (!fin.is_open()) {
            cerr << "Файл не может быть открыт";
            return 1;
      }
      while (!fin.eof()) {
//          fin >> buff;
            fin.getline(buff,128);
            cmd=atoi(buff);
            if (cmd>0) {
            gf->analyzeCom(cmd);
            }
      }
      fin.close();
      return 0;
}
```

Файл с командами. Коды команд выбираются самостоятельно.

```
Команды для курсового проекта
  секция проверки конфигурации
 8  запрос дедушки о его бабушке
 9  запрос бабушки о ее дедушке
10  запрос курочки о ее хозяевах
11  запрос мышки, где она живет
  секция команд реализации сюжета сказки
5  Курочка несет яйцо
  1   Дедушка бьет яйцо
 3    Бабушка бьет яйцо
   7  Мышка бежала хвостиком махнула разбила яйцо
 2    Дедушка плачет
  4   Бабушка плачет
     6 Курочка успокаивает
```