**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

**Кафедра информатики и прикладной математики**

Программирование Интернет-Приложений

Лабораторная работа №2

Выполнил: Гхази Даниэль

Группа P3218

Преподаватель: Гаврилов А.В.

2016 г.

**1. Задание**

Скомпилировать и запустить исходный код на языке программирования Java, выданный в соответствии с вариантом. Разобраться том, как реализуются принципы объектно-ориентированного программирования в получившейся программе, и том, почему она выдаёт такой результат. Добавить комментарии в ключевые фрагменты программы, поясняющие её поведение.

**2. Код получившейся программы**

*// Вариант 1959***public class** Lab2 {  
 **public static void** main(String[] args) {  
 Remoraid mother = **new** Remoraid();  
 Octillery daughter = **new** Octillery();  
 *// Экземпляр класса Remoraid "brother" содержит ссылку на новый объект класса-наследника Octillery* Remoraid brother = **new** Octillery();  
  
 mother.*sniper*();  
 *// Вызов перегруженного метода defenseCurl класса Remoraid, т.к. mother хранит ссылку на объект класса Remoraid.  
 // Конкретный перегрузка выбирается в зависимости от типа передаваемого объекта, в данном случае - Remoraid* mother.defenseCurl(brother);  
 daughter.conversion();  
 *// Т.к. daughter является объектом класса Octillery, будет вызвана реализация метода focusEnergy() из  
 // класса-наследника Octillery* daughter.focusEnergy();  
 *// brother может вызвать метод moody() класса Octillery, т.к. хранит ссылку на объект тип этого класса, хоть он  
 // сам и является объектом класса Remoraid* brother.moody();  
 *// Т.к. brother хранит в себе ссылку на объект типа Octillery, будет вызвана реализация метода focusEnergy()  
 // именно из класса-наследника Octillery, хоть brother и является объектом класса Remoraid* brother.focusEnergy();  
 daughter.swordsDance();  
 *// Вызывается реализация метода sniper() из базового класса, хоть brother и хранит ссылку на объект класса  
 // Octillery. Это происходит потому, что метод sniper() имеет модификатор static. Это приводит к тому, что  
 // выбор вызываемого метода происходит на этапе компиляции, а не во время исполнения программы* brother.*sniper*();  
 daughter.defenseCurl(mother);  
 mother.defenseCurl(daughter);  
 *// Вызов перегруженного метода defenseCurl класса Octillery, т.к. daughter хранит ссылку на объект класса  
 // Octillery. Конкретный перегрузка выбирается в зависимости от типа передаваемого объекта, в данном случае -  
 // Remoraid, т.к. brother является объектом класса Remoraid.* daughter.defenseCurl(brother);  
 daughter.sharpen();  
 daughter.*sniper*();  
 mother.workUp();  
 mother.focusEnergy();  
 *// Приводим экземпляр класса Remoraid brother к типу класса-наследника Octillery, чтобы появилась возможность  
 // вызывать метод этого класса batonPass()* ((Octillery)brother).batonPass();  
 }  
}  
  
**class** Remoraid {  
 **byte jump**;  
 **protected** String **psychic** = **"Psychic"**;  
 **double accuracy** = 7.3;  
 **public int troublesome**;  
 **protected** String **shadow** = **"Shadow"**;  
 **protected** String **psychicShadow** = **"PsychicShadow"**;  
 **protected static int** *inflatable* = 99;  
  
 **public** Remoraid() {  
 **jump** = (**byte**) 0x99;  
 }  
  
 {  
 **jump** = 57;  
 **troublesome** = 060;  
 }  
  
  
 **public void** defenseCurl(Octillery p) {  
 System.***out***.println(**"Remoraid attacks Octillery with Defense Curl"**);  
 }  
  
 **public static void** sniper() {  
 System.***out***.println(**"Remoraid casts Sniper"**);  
 }  
  
 **public void** defenseCurl(Remoraid p) {  
 System.***out***.println(**"Remoraid attacks Remoraid with Defense Curl"**);  
 }  
  
 **public void** focusEnergy() {  
 System.***out***.println(**"Remoraid casts Focus Energy"**);  
 }  
  
 **public void** workUp() {  
 **double** defense = 5.2;  
  
 *// Выведет false, т.к. результат разности float-переменных неточен* System.***out***.println((**accuracy** - defense) == 2.1);  
 }  
  
 **public void** sharpen() {  
 System.***out***.println(*inflatable* + **jump**);  
 System.***out***.println(**troublesome** - *inflatable*);  
 System.***out***.println(**jump** - **troublesome**);  
 }  
  
 **public void** moody() {  
 *// Здесь equals() сравнивает значения строк, а оператор '==' - ссылки* System.***out***.println(**psychicShadow**.equals(**"Psychic"**+**"Shadow"**));  
 System.***out***.println(**psychicShadow**.equals(**psychic**+**shadow**));  
 System.***out***.println(**psychicShadow** == **psychic**+**shadow**);  
 *// Получаем true, т.к. строка конкатенация строк "Psychic" + "Shadow" происходит на этапе компиляции и уже  
 // готовая строка помещается в пул строк. psychicShadow также ссылается на эту строку в пуле строк,  
 // следовательно они указывают на один и тот же объект, следовательно возвращается true* System.***out***.println(**psychicShadow** == **"Psychic"**+**"Shadow"**);  
 System.***out***.println(**psychicShadow**.equals(**psychic**+**"Shadow"**));  
 System.***out***.println(**psychicShadow** == **"Psychic"**+**shadow**);  
 }  
}  
  
**class** Octillery **extends** Remoraid {  
 **private int stealth**;  
 **private** String **flyingShadow** = **"FlyingShadow"**;  
 **float depth** = 5.3f;  
 **private** String **flying** = **"Flying"**;  
  
 **public** Octillery() {  
 **stealth** = 99;  
 }  
  
  
 **public void** focusEnergy() {  
 System.***out***.println(**"Octillery casts Focus Energy"**);  
 }  
  
 **public void** conversion() {  
 System.***out***.println(*inflatable* - **stealth**);  
 System.***out***.println(**stealth** - **jump**);  
 System.***out***.println(**stealth** + **troublesome**);  
 }  
  
 **public void** swordsDance() {  
 **float** length = 6.2f;  
  
 *// Выведет false, т.к. результат разности float-переменных неточен* System.***out***.println((length - **depth**) == 0.9f);  
 }  
  
 **public static void** sniper() {  
 System.***out***.println(**"Octillery casts Sniper"**);  
 }  
  
 **public void** batonPass() {  
 System.***out***.println(**flyingShadow** == **flying**+**shadow**);  
 System.***out***.println(**flyingShadow** == **new** String(**"Flying"**+**"Shadow"**));  
 System.***out***.println(**flyingShadow** == **new** String(**"FlyingShadow"**));  
 *// Получаем true, т.к. мы итенировали строку, содержащуююся в flying+shadow, в пул строк и, затем, сравнили две  
 // ссылки на одну и ту же строку в пуле строк* System.***out***.println(**flyingShadow** == (**flying**+**shadow**).intern());  
 }  
  
 **public void** defenseCurl(Octillery p) {  
 System.***out***.println(**"Octillery attacks Octillery with Defense Curl"**);  
 }  
  
 **public void** defenseCurl(Remoraid p) {  
 System.***out***.println(**"Octillery attacks Remoraid with Defense Curl"**);  
 }  
}

**3. Результаты выполнения**

Remoraid casts Sniper

Remoraid attacks Remoraid with Defense Curl

0

202

147

Octillery casts Focus Energy

true

true

false

true

true

false

Octillery casts Focus Energy

false

Remoraid casts Sniper

Octillery attacks Remoraid with Defense Curl

Remoraid attacks Octillery with Defense Curl

Octillery attacks Remoraid with Defense Curl

-4

-51

-151

Octillery casts Sniper

false

Remoraid casts Focus Energy

false

false

false

true

Process finished with exit code 0

**4. Вывод**

В ходе работы мною было изучено создание объектов пользовательских типов, вызов их методов, наследование классов и особенности работы таких классов, явное приведение типов, особенности операций с числами с плавающей точкой, сравнение строк, а также различные модификаторы. Полученные знания являются существенными для написания программ написание разноплановых программ на Java.