**Лабораторная работа №2.**

Тема: Объективно-ориентированное программирование и диаграмма классов

Цель работы: освоить навыки использования методов ООП: инкапсуляция, полиформизм, наследование

1. **Индивидуальные задания**

1. Напишите программу с классом Mother от которого наследуется класс Daughter.

Сделайте так, чтобы результат print(object) был разный.

Воспользуйтесь принципами полиморфизма, наследования и инкапсуляции.

public class Main{  
 public static void main(String[] args) {  
 Mother mother = new Mother("Анна");  
 Daughter daughter= new Daughter("Аяна");  
 mother.print();  
 daughter.print();  
 }  
}

public class Mother {  
  
 public Mother (String n) {  
 setName(n);  
 };  
private String name;  
public String getName() {  
 return name;  
}  
  
public void setName(String name) {  
 this.name=name;}  
 void print(){  
 System.*out*.println("Мама"+getName());  
 }  
  
 }

public class Daughter extends Mother {  
  
  
 public Daughter(String n) {  
 super(n);  
 }  
  
 @Override  
 void print(){  
 System.*out*.println("Дочка"+getName());  
  
 }  
}

2. Реализйте класс Animal. Внутри объявите поле для имени и возраста.

От класса Animal унаследуйте класс Zebra и Dolphin.

Оба класса могут вернуть описание, содержащее имя, возраст и какую-то доп.информацию, например, что это за вид животного.

Воспользуйтесь принципами полиморфизма, наследования и инкапсуляции.

public class Main {  
 public static void main(String[] args) {  
Animal Zebra = new Zebra("Арчи",14);  
Animal Dolphin = new Dolphin("Альфа",5);  
System.*out*.println(Zebra.getDescription());  
 System.*out*.println(Dolphin.getDescription());  
 }  
 }

public class Animal {  
 private String name;  
 private int age;  
 public Animal( String name, int age){  
 this.name= name ;  
 this.age= age;  
 }  
 public int getAge(){  
 return age;  
 }  
 public String getDescription(){  
 return "Имя: "+ name + ", возраст"+ age;  
 }  
}

public class Zebra extends Animal{  
 public Zebra(String name, int age){  
 super(name, age);  
 }  
 @Override  
 public String getDescription(){  
 return super.getDescription()+ ", вид: зебра";  
 }  
}

public class Dolphin extends Animal {  
public Dolphin(String name, int age){  
 super(name, age);  
}  
@Override  
 public String getDescription(){  
 return super.getDescription()+"вид: дельфин";  
}  
}

3. Игра "Арифметические драконы" предназначена для обучения детей арифметике. На героя нападает дракон, который задаёт вопрос на сложение (если дракон зелёный), вычитание (красный) или умножение (чёрный).

Реализуйте следующие классы:

