ЛАБОРАТОРНАЯ РАБОТА 4

НАСЛЕДОВАНИЕ РЕАЛИЗАЦИИ

**Цель** — изучить принципы механизма наследования реализации при разработке объектно-ориентированных приложений.

**Задачи:**

**—** изучить синтаксические принципы реализации наследования;

— изучить принципы реализации наследования при решении практических задач.

ВАРИАНТ 12

Класс TechnicalMean:

package v\_12;  
  
public abstract class TechnicalMean {  
 protected String ownerName;  
 protected int fromYear;  
  
 public abstract void go();  
  
 public TechnicalMean(String ownerName, int fromYear) {  
 this.ownerName = ownerName;  
 this.fromYear = fromYear;  
 }  
  
 @Override  
 public String toString() {  
 return "TecknicalMean of " + ownerName + " from " + fromYear;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof TechnicalMean)) return false;  
 TechnicalMean technicalMean = (TechnicalMean) object;  
 return (this.fromYear == technicalMean.fromYear) && (this.ownerName.equals(technicalMean.ownerName));  
 }  
}

Класс Car extends TechnicalMean:

package v\_12;  
  
public class Car extends TechnicalMean {  
 @Override  
 public void go() {  
 System.*out*.println("Поехал");  
 }  
  
 public Car(String ownerName, int fromYear) {  
 super(ownerName, fromYear);  
 }  
  
 public String toString() {  
 return "Car of " + ownerName + " from " + fromYear;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof Car)) return false;  
 Car car = (Car) object;  
 return (this.fromYear == car.fromYear) && (this.ownerName.equals(car.ownerName));  
 }  
}

Класс PassengerCar extends Car:

package v\_12;  
  
import java.util.Arrays;  
  
public final class PassengerCar extends Car {  
 private String[] passangerNames;  
  
 public PassengerCar(String ownerName, int fromYear, String[] passangerNames) {  
 super(ownerName, fromYear);  
 this.passangerNames = passangerNames;  
 }  
  
 private String getPassengers() {  
 StringBuilder passengers = new StringBuilder();  
 for (String passenger : passangerNames) passengers.append(passenger).append(" ");  
 return passengers.toString();  
 }  
  
 @Override  
 public String toString() {  
 return "Passenger car of " + ownerName + " from " + fromYear + " with " + getPassengers();  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof PassengerCar)) return false;  
 PassengerCar passengerCar = (PassengerCar) object;  
 return (this.fromYear == passengerCar.fromYear) && (this.ownerName.equals(passengerCar.ownerName));  
 }  
}

Класс Truck extends Car:

package v\_12;  
  
import java.util.Arrays;  
  
public final class Truck extends Car {  
 private double loadCapacity;  
  
 public Truck(String ownerName, int fromYear, double loadCapacity) {  
 super(ownerName, fromYear);  
 this.loadCapacity = loadCapacity;  
 }  
  
 @Override  
 public String toString() {  
 return "Car of " + ownerName + " from " + fromYear + " with load capacity = " + loadCapacity;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof Truck)) return false;  
 Truck truck = (Truck) object;  
 return (this.fromYear == truck.fromYear) && (this.ownerName.equals(truck.ownerName));  
 }  
}

Класс AirTransport extends TechnicalMean:

package v\_12;  
  
public abstract class AirTransport extends TechnicalMean {  
 protected double maxHeight;  
  
 public AirTransport(String ownerName, int fromYear, double maxHeight) {  
 super(ownerName, fromYear);  
 this.maxHeight = maxHeight;  
 }  
  
 @Override  
 public void go() {  
 System.*out*.println("Взлетел");  
 }  
  
 @Override  
 public String toString() {  
 return "AirTransport of " + ownerName + " from " + fromYear + " with max height " + maxHeight;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof AirTransport)) return false;  
 AirTransport airTransport = (AirTransport) object;  
 return (this.fromYear == airTransport.fromYear) && (this.ownerName.equals(airTransport.ownerName)) &&  
 (this.maxHeight == airTransport.maxHeight);  
 }  
}

Класс Airplane extends AirTransport:

package v\_12;  
  
public class Airplane extends AirTransport {  
 public Airplane(String ownerName, int fromYear, double maxHeight) {  
 super(ownerName, fromYear, maxHeight);  
 }  
  
 @Override  
 public String toString() {  
 return "Airplane of " + ownerName + " from " + fromYear + " with max height " + maxHeight;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof Airplane)) return false;  
 Airplane airplane = (Airplane) object;  
 return (this.fromYear == airplane.fromYear) && (this.ownerName.equals(airplane.ownerName)) &&  
 (this.maxHeight == airplane.maxHeight);  
 }  
}

Класс PassengerAirplane extends Airplane:

package v\_12;  
  
public final class PassengerAirplane extends Airplane {  
 private int kolPassengers;  
  
 public PassengerAirplane(String ownerName, int fromYear, double maxHeight, int kolPassengers) {  
 super(ownerName, fromYear, maxHeight);  
 this.kolPassengers = kolPassengers;  
 }  
  
 @Override  
 public String toString() {  
 return "Passenger airplane of " + ownerName + " from " + fromYear + " with max height " + maxHeight +  
 " and count of passengers = " + kolPassengers;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof PassengerAirplane)) return false;  
 PassengerAirplane passengerAirplane = (PassengerAirplane) object;  
 return (this.fromYear == passengerAirplane.fromYear) && (this.ownerName.equals(passengerAirplane.ownerName)) &&  
 (this.maxHeight == passengerAirplane.maxHeight) && (this.kolPassengers == passengerAirplane.kolPassengers);  
 }  
}

Класс Helicopter extends AirTransport:

package v\_12;  
  
public final class Helicopter extends AirTransport {  
 public Helicopter(String ownerName, int fromYear, double maxHeight) {  
 super(ownerName, fromYear, maxHeight);  
 }  
  
 @Override  
 public String toString() {  
 return "Helicopter of " + ownerName + " from " + fromYear + " with max height " + maxHeight;  
 }  
  
 @Override  
 public boolean equals(Object object) {  
 if (this == object) return true;  
 if (object == null) return false;  
 if (!(object instanceof Helicopter)) return false;  
 Helicopter helicopter = (Helicopter) object;  
 return (this.fromYear == helicopter.fromYear) && (this.ownerName.equals(helicopter.ownerName)) &&  
 (this.maxHeight == helicopter.maxHeight);  
 }  
}

Класс Main:

package v\_12;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Main {  
  
 public static void main(String[] args) {  
 List<TechnicalMean> technicalMeans= new ArrayList<>();  
 technicalMeans.add(new Airplane("Airflot", 1999, 290));  
 technicalMeans.add(new Car("Mark Sizov", 2017));  
 technicalMeans.add(new Helicopter("GrayDev", 2018, 560));  
 technicalMeans.add(new PassengerAirplane("Ural airlines", 2004, 430,  
 500));  
 technicalMeans.add(new PassengerCar("Viktor Eliseev", 2021,  
 new String[]{"Ann", "Ingrid"}));  
 technicalMeans.add(new Truck("Belerusss", 2014, 2500));  
  
 for (TechnicalMean technicalMean : technicalMeans){  
 technicalMean.go();  
 System.*out*.println(technicalMean);  
 }  
  
 PassengerCar c1 = new PassengerCar("Nelson", 2000, new String[]{"Inga"});  
 Car c2 = new PassengerCar("Nelson", 2000, new String[]{"Inga"});  
 Truck c3 = new Truck("Nelson", 2000, 500);  
 Car c4 = new Truck("Nelson", 2000, 500);  
 System.*out*.println(c1.equals(c2) + " " + c2.equals(c1));  
 System.*out*.println(c2.equals(c3) + " " + c3.equals(c2));  
 System.*out*.println(c3.equals(c4) + " " + c4.equals(c3));  
 System.*out*.println(c2.equals(c4) + " " + c4.equals(c2));  
 }  
}

**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1) При наследовании реализации также наследуются поля и методы класса-родителя, например, в классе B есть массив bb и метод, который вычисляет сумму элементов массива bb, тогда класс A, унаследованный от класса B тоже будет иметь эти массив и метод, включая реализацию метода.

Изображение выглядит как текст, антенна

Автоматически созданное описание2) Конструкция для реализации наследования в java – extends, например, class A extends B{} из предыдущего ответа.

3) Полиморфизм в java – возможность переопределения метода в наследниках. Например, имеются классы Rectangle, Triangle, Circle, а также объекты данных типов. Чтобы не перебирать вручную все объекты, можно сложить их в массив такого типа, от которого унаследуются все фигуры, например, Figure. Создадим метод в классе Figure и переопределим его в остальных. Таким образом при вызове данного метода на каждом из элементов массива, будет выполняться переопределенный метод.

4) Суперкласс – класс, от которого унаследованы другие классы.

Подкласс – класс, который наследуется от другого класса.

Наследование реализации – наследование реализации метода суперкласса подклассом.

Класс-наследник – то же, что и подкласс.

Иерархическая цепочка – дерево, элементами которого являются классы и интерфейсы.

Динамическое связывание – позднее связывание, происходит во время выполнения программы с использованием конкретных объектов.

5) Класс Object имеет методы:

toString() – метод, переводящий объект в строку,

equals() – проверка двух элементов на равенство,

hashCode() – позволяет задать элементу некоторое числовое значение(хэш-еод),

getClass() – позволяет получить тип данного объекта.

6) Переопределение метода.