JavaClass23 Method Overriding

Method Overriding:

Method overriding allows us to have same methods in parent and child classes.

Why we should override a method?

1) when we inherit from a class everything other than private methods and constructors is inherited if we

don't like any method from the parent we can override.

2) it helps us achieve polymorphism.

what is the syntax to override a method:

1) First there must be a parent child relationship.

2) In the child class just redefine the method.

Type casting in objects:

As we can type cast in the primitive data types we can also type cast objects.

like storing the object of a child class in a variable of parent type.

Why we should type cast objects?

with the help of type casting we can start storing the objects of child classes in an array of type parent.

what is the syntax to type cast?

its same as primitive type caseting for example if we have an animal class and animal class has a child class Cat

we can do

Animal animal=new Cat();

parent name[] arr={new childclass name()}

polymorphism:

polymorphism is the ability of a method to take on many forms based on the object or the parameters that we pass

to the method.

Types of polymorphism:

There are two types of polymorphism. 1) compile time/static/early binding or run time/dynamic or late binding.

compile time/static/early binding:

When we use method overloading to achieve polymorphism it is called compile time polymorphism.

Run time/dynamic or late binding:

When we use method overriding to achieve polymorphism then it is called Run time polymorphism.

final Keyword:

final keyword is just like any other keyword in java like class package void etc.

There are three man uses of final keyword.

1) If we use final keyword with a class no other class can inherit from that class.

2) If we use it with a method name no child class can override that method.

3) If we use it a field once the value has been assigned to that field no one can change it value afterwards.

package class23;

public abstract class Animal {

abstract void eat();

abstract void sleep();

void walk(){

System.out.println("Cat walk");

}

public static void main(String[] args) {

//Animal animal=new Animal() ;

// aniaml.eat();

// Break till 11:50

}

}

class Cat extends Animal{

@Override

void eat() {

System.out.println("Cats like fish");

}

void sleep(){

System.out.println("Cats like to sleep 12 to 16 hours");

}

void showAttitude(){

System.out.println("hmm hmm hmm hmm");

}

}

class Dog extends Animal{

@Override

void eat() {

System.out.println("Cats like watermelon");

}

void sleep(){

System.out.println("Cats like to sleep 8 to 10 hours");

}

}

package class23;

public abstract class DataBase {

abstract void openDatabase();

abstract void readData();

abstract void writeData();

abstract void closeDatabase();

}

/\*

Hover over the error and click on the implement methods option

\*/

class MicrosoftDataBase extends DataBase{

@Override

void openDatabase() {

System.out.println("Opening the Microsoft Database");

}

@Override

void readData() {

System.out.println("Reading the data from MSDatabase");

}

@Override

void writeData() {

System.out.println("writing the data to MSDataBase");

}

@Override

void closeDatabase() {

System.out.println("Closing the MSDatabase");

}

}

class Oracle extends DataBase{

@Override

void openDatabase() {

System.out.println("Opening the Oracle database");

}

@Override

void readData() {

System.out.println("Reading the data from Oracle databse");

}

@Override

void writeData() {

System.out.println("Writing the data to Oracle database");

}

@Override

void closeDatabase() {

System.out.println("Closing the Oracle database");

}

}

package class23;

public class DataBaseTester {

public static void main(String[] args) {

DataBase dataBase=new Oracle();

dataBase.openDatabase();

dataBase.readData();

dataBase.writeData();

dataBase.closeDatabase();

}

}

package class23;

public interface IDataBase {

void openDatabase();

void readData();

void writeData();

}

class IBM implements IDataBase{

@Override

public void openDatabase() {

System.out.println("Opening the IBM database");

}

@Override

public void readData() {

System.out.println("Reading the data from the IBM Database");

}

@Override

public void writeData() {

System.out.println("Writing the data to the database");

}

}

package class23;

public class InterfaceTester {

public static void main(String[] args) {

IDataBase iDataBase=new IBM();

iDataBase.openDatabase();

iDataBase.readData();

iDataBase.writeData();

//1:40

}

}

package class23;

public interface Person {

void eat();

}

interface Employee{

void work();

}

interface SyntaxEmployee extends Employee,Person{

void creatRepl();

}

class Asghar implements SyntaxEmployee{

@Override

public void eat() {

System.out.println("Like to eat grilled Fish");

}

@Override

public void work() {

System.out.println("Teaches Bacth14 ");

}

@Override

public void creatRepl() {

System.out.println("Create easy Repls");

}

}

package class23;

public class StudentTester {

public static void main(String[] args) {

Student[] arr={new CollegeStudent(),

new SchoolStudent(),new SyntaxStudent()};

for (Student st:arr

) {

st.study();

st.doHomeWork();

st.doPractice();

}

}

}

package class23;

public class Task1 {

/\*

Create a Class Student that will have 3 subclasses as SyntaxStudent, CollegeStudent, SchoolStudent.

Define common behavior within parent class and override some of the methods in child classes

Define some methods specific to child classes.

Write example of achieving run time polymorphism.

\*/

}

class Student{

void study(){

System.out.println("students must study");

}

void doHomeWork(){

System.out.println("Students must solve their homeworks");

}

void doPractice(){

System.out.println("Students must practice to get good marks");

}

}

class SyntaxStudent extends Student{

@Override

void doHomeWork() {

System.out.println("Syntax students must solve all Repls three time ");

}

@Override

void doPractice() {

System.out.println("Syntax student must do their own research and practice ");

}

}

class CollegeStudent extends Student{

@Override

void doPractice() {

System.out.println("College students must practice to get good marks");

}

}

class SchoolStudent extends Student{

}