Inheritance: super class and their sub class.

Overloaded constructor:

package com.company;

class Base1{

Base1(){

System.out.println("I am a constructor");

}

Base1(int x){

System.out.println("I am an overloaded constructor with value of x as: " + x);

}

}

class Derived1 extends Base1{

Derived1(){

//super(0);

System.out.println("I am a derived class constructor");

}

Derived1(int x, int y){

super(x);

System.out.println("I am an overloaded constructor of Derived with value of y as: " + y);

}

}

class ChildOfDerived extends Derived1{

ChildOfDerived(){

System.out.println("I am a child of derived constructor");

}

ChildOfDerived(int x, int y, int z){

super(x, y);

System.out.println("I am an overloaded constructor of Derived with value of z as: " + z);

}

}

public class cwh\_46\_constructors\_in\_inheritance {

public static void main(String[] args) {

// Base1 b = new Base1();

// Derived1 d = new Derived1();

// Derived1 d = new Derived1(14, 9);

// ChildOfDerived cd = new ChildOfDerived();

ChildOfDerived cd = new ChildOfDerived(12, 13, 15);

}

}

Over Riding :

package com.company;

class A{

public int a;

public int harry(){

return 4;

}

public void meth2(){

System.out.println("I am method 2 of class A");

}

}

class B extends A{

@Override

public void meth2(){

System.out.println("I am method 2 of class B");

}

public void meth3(){

System.out.println("I am method 3 of class B");

}

}

public class cwh\_48\_method\_overriding {

public static void main(String[] args) {

A a = new A();

a.meth2();

B b = new B();

b.meth2();

}

}

NOTE :

“IF BOTH METHODS ARE SAME IN PARENT AND CHILD CLASS BUT IT IS STATIC , CALLED MOTHOD HIDING.”

ABSTARCT CLASS AND METHOD :

package com.company;

abstract class Parent2{

public Parent2(){

System.out.println("Mai base2 ka constructor hoon");

}

public void sayHello(){

System.out.println("Hello");

}

abstract public void greet();

abstract public void greet2();

}

class Child2 extends Parent2{

@Override

public void greet(){

System.out.println("Good morning");

}

@Override

public void greet2(){

System.out.println("Good afternoon");

}

}

abstract class Child3 extends Parent2{

public void th(){

System.out.println("I am good");

}

}

public class cwh\_53\_abstract {

public static void main(String[] args) {

//Parent2 p = new Parent2(); -- error

Child2 c = new Child2();

//Child3 c3 = new Child3(); -- error

}

}

INTERFACES :

Example 1:

interface Bicycle {

void apply brake ( int decrement );

void speed up ( int increment );

}

class Avon cycle implements Bicycle {

int speed = 7 ;

void apply brake ( int decrement ) {

speed = speed - decrement ;

}

void speedup ( int increment ){

speed = speed + increment ;

}

Example 2:

package com.company;

interface Bicycle{

int a = 45;

void applyBrake(int decrement);

void speedUp(int increment);

}

interface HornBicycle{

int x = 45;

void blowHornK3g();

void blowHornmhn();

}

class AvonCycle implements Bicycle, HornBicycle{

//public int x = 5;

void blowHorn(){

System.out.println("Pee Pee Poo Poo");

}

public void applyBrake(int decrement){

System.out.println("Applying Brake");

}

public void speedUp(int increment){

System.out.println("Applying SpeedUP");

}

public void blowHornK3g(){

System.out.println("Kabhi khushi kabhi gum pee pee pee pee");

}

public void blowHornmhn(){

System.out.println("Main hoon naa po po po po");

}

}

public class cwh\_54\_interfaces {

public static void main(String[] args) {

AvonCycle cycleHarry = new AvonCycle();

cycleHarry.applyBrake(1);

// You can create properties in Interfaces

System.out.println(cycleHarry.a);

System.out.println(cycleHarry.x);

// You cannot modify the properties in Interfaces as they are final

// cycleHarry.a = 454;

//System.out.println(cycleHarry.a);

cycleHarry.blowHornK3g();

cycleHarry.blowHornmhn();

}

}