1. Code this:

public abstract class Course

{

public void start()

{

// start process

}

public abstract double getFee();

public abstract void evaluate();

}

define child classes of above abstract class

DACDBDACourse extends Course

MSCitCourse extends Course

show late binding with the help of "perform()" method.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | **public** **abstract** **class** Course {  **public** **void** start()  {  System.***out***.println("start process");  }  **public** **abstract** **double** getFee();  **public** **abstract** **void** evaluate();  } | | **public** **class** DACDBDACourse **extends** Course {  **private** **double** fee;    DACDBDACourse() {  fee=1000000;  }  **public** **double** getFee() {  // **TODO** Auto-generated method stub  **return** fee;  }  **public** **void** evaluate() {  // **TODO** Auto-generated method stub  System.***out***.println("evaluating DAC/DBDA Course");  }  } | | **public** **class** MSCitCourse **extends** Course {  **private** **double** fee;    MSCitCourse() {  fee=50000;  }  **public** **double** getFee() {  // **TODO** Auto-generated method stub  **return** fee;  }  **public** **void** evaluate() {  // **TODO** Auto-generated method stub  System.***out***.println("evaluating MSCIT Course");  }  } | | **public** **class** CourseDemo {  **public** **static** **void** perform(Course c) {  c.start();  System.***out***.println("Rs. "+c.getFee());  c.evaluate();  }    **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  *perform*(**new** DACDBDACourse());  System.***out***.println();  *perform*(**new** MSCitCourse());  }  } | |
|  |
| start process  Rs. 1000000.0  evaluating DAC/DBDA Course  start process  Rs. 50000.0  evaluating MSCIT Course |

1. Code this:

abstract class Beverages {

// Defined

public void addMilk() {

}

// abstract

abstract void addIngredients();

}

class Coffee extends Beverages

{

void addIngredients()

{

// ingredients for Coffee

}

// addMilk inherited

}

class Tea extends Beverages

{

void addIngredients()

{

// ingredients for Tea

}

// addMilk inherited

}

class FruitJuice extends Beverages

{

void addIngredients()

{

// ingredients for FruitJuice

}

// addMilk inherited

}

create Array of Beverages class and store all the child classes objects.

now traverse the array and invoke all the functions.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **public** **abstract** **class** Beverages {  // Defined  **public** **void** addMilk() {  System.***out***.println("adding milk");  }  // abstract  **abstract** **void** addIngredients();  } | | **public** **class** Coffee **extends** Beverages {  **void** addIngredients() {  // **TODO** Auto-generated method stub  System.***out***.println("adding coffee ingredients");  }    } | | **public** **class** Tea **extends** Beverages {  **void** addIngredients() {  // **TODO** Auto-generated method stub  System.***out***.println("adding tea ingredients");  }  } | | **public** **class** FruitJuice **extends** Beverages{  **void** addIngredients() {  // **TODO** Auto-generated method stub  System.***out***.println("adding Fruit juice ingredients");  }  } | | **public** **class** BeveragesDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Beverages bev[] = **new** Beverages[3];  bev[0] = **new** Coffee();  bev[1] = **new** Tea();  bev[2] = **new** FruitJuice();    **for**(**int** i=0; i<bev.length; i++) {  bev[i].addMilk();  bev[i].addIngredients();  System.***out***.println();  }  }  } | |
|  |
| adding milk  adding coffee ingredients  adding milk  adding tea ingredients  adding milk  adding Fruit juice ingredients |

1. Code this:

create abstract class "Shape" with

a contract "draw()" and concrete behaviour "show()"

create 3 child classes

Triangle, Rectangle and Circle

define a class "ShapeDemo" with main. inside main create an array of "Shape" of 3 elements. Store all the child class objects inside the array, traverse it and invoke "draw()" and "show()" methods polymorphically.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **public** **abstract** **class** Shape {  **public** **void** show() {  System.***out***.println("showing "+**this**.getClass().getName());  }  **abstract** **void** draw();  } | | **public** **class** Triangle **extends** Shape {  **void** draw() {  // **TODO** Auto-generated method stub  System.***out***.println("drawing Triangle");  }    } | | **public** **class** Rectangle **extends** Shape {  **void** draw() {  // **TODO** Auto-generated method stub  System.***out***.println("drawing Rectangle");  }  } | | **public** **class** Circle **extends** Shape{  **void** draw() {  // **TODO** Auto-generated method stub  System.***out***.println("drawing Circle");  }  } | | **public** **class** ShapeDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Shape s[] = **new** Shape[3];  s[0] = **new** Triangle();  s[1] = **new** Rectangle();  s[2] = **new** Circle();    **for**(**int** i=0; i<s.length; i++) {  s[i].draw();  s[i].show();  System.***out***.println();  }  }  } | |
|  |
| drawing Triangle  showing Triangle  drawing Rectangle  showing Rectangle  drawing Circle  showing Circle |

1. Code this:

public class UIComponent

{

}

public class Button extends UIComponent

{

}

public class TextField extends UIComponent

{

}

public class CheckBox extends UIComponent

{

}

public abstract class UIComponentCreator

{

public void show(String type)

{

UIComponent comp=createUIComponent(type);

add(comp);

}

public void add(UIComponent component)

{

System.out.println("Added the component\t"+component);

}

public abstract UIComponent createUIComponent(String type);

}

create 3 child classes of "UIComponentCreator"

WindowsUIComponentCreator

LinuxUIComponentCreator

MacUIComponentCreator

in all these classes "createUIComponent()" method must be defined in such a way that whatever type is passed as per this it must return a specific UIComponent. e.g., if "button" is passed, it should return "Button" class object or if "textfield" is passed, it should return "TextField" class object.

create a class UIComponentCreatorDemo with main function

inside main function. create objects of WindowsUIComponentCreator, LinuxUIComponentCreator and MacUIComponentCreator classes and invoke "createUIComponent ()”,"add ()" and "show ()" methods.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **public** **class** UIComponent {    } | | **public** **class** Button **extends** UIComponent {  } | | **public** **class** TextField **extends** UIComponent {  } | | **public** **class** CheckBox **extends** UIComponent{  } | | **public** **abstract** **class** UIComponentCreator {    **public** **void** show(String type)  {  UIComponent comp = createUIComponent(type);  add(comp);  }    **public** **void** add(UIComponent component)  {  System.***out***.println("Added the component\t"+component.getClass().getName());  }    **public** **abstract** UIComponent createUIComponent(String type);  }  **class** WindowsUIComponentCreator **extends** UIComponentCreator {  **public** UIComponent createUIComponent(String type) {  // **TODO** Auto-generated method stub  **switch**(type.toLowerCase()) {  **case** "button": **return** **new** Button();  **case** "textfield": **return** **new** TextField();  **case** "checkbox": **return** **new** CheckBox();  **default** : **return** **null**;  }  }  }  **class** LinuxUIComponentCreator **extends** UIComponentCreator {  **public** UIComponent createUIComponent(String type) {  // **TODO** Auto-generated method stub  **switch**(type.toLowerCase()) {  **case** "button": **return** **new** Button();  **case** "textfield": **return** **new** TextField();  **case** "checkbox": **return** **new** CheckBox();  **default** : **return** **null**;  }  }  }  **class** MacUIComponentCreator **extends** UIComponentCreator {  **public** UIComponent createUIComponent(String type) {  // **TODO** Auto-generated method stub  **switch**(type.toLowerCase()) {  **case** "button": **return** **new** Button();  **case** "textfield": **return** **new** TextField();  **case** "checkbox": **return** **new** CheckBox();  **default** : **return** **null**;  }  }  } | | **public** **class** UIComponentCreatorDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub    UIComponentCreator ui[] = **new** UIComponentCreator[3];    ui[0] = **new** WindowsUIComponentCreator();  ui[1] = **new** LinuxUIComponentCreator();  ui[2] = **new** MacUIComponentCreator();    String components[] = {"button", "TextField", "CheckBox"};    **for**(**int** i=0; i<ui.length; i++) {  ui[i].show(components[i]);  System.***out***.println();  }  }  } | |
|  |
| Added the component Button  Added the component TextField  Added the component CheckBox |

1. accept 3 numbers through command line arguments and display their sum.

|  |
| --- |
| **public** **class** CommandLineDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub    **int** sum=0;  **for**(**int** i=0;i<args.length;i++)  sum+=Integer.*parseInt*(args[i]);  System.***out***.println(sum);  }  } |
|  |
| C:\Users\kunal\OneDrive\Desktop>javac CommandLineDemo.java  C:\Users\kunal\OneDrive\Desktop>java CommandLineDemo 10 25 15  50  C:\Users\kunal\OneDrive\Desktop> |

1. Code this:

on developer side:

create interface "Shape" with "void draw();" method.

now define child classes

Triangle,Rect and Circle

on Client side:

define "ShapeDemo" class with main and perform functions

define perform function in such a way that given any shape it can invoke "draw()" of it.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dev side   |  | | --- | | /\*\*  \*  \*/  package devpack;  /\*\*  \* @author Kunal  \*  \*/  public interface Shape {  void draw();  } | | /\*\*  \*  \*/  package devpack;  /\*\*  \* @author Kunal  \*  \*/  public class Triangle implements Shape {  public void draw() {  System.out.println("drawing triangle");  }  } | | /\*\*  \*  \*/  package devpack;  /\*\*  \* @author Kunal  \*  \*/  public class Rectangle implements Shape {  public void draw() {  System.out.println("drawing rectangle");  }  } | | /\*\*  \*  \*/  **package** devpack;  /\*\*  \* **@author** Kunal  \*  \*/  **public** **class** Circle **implements** Shape {  **public** **void** draw() {  System.***out***.println("drawing circle");  }  } | |
| **//client side**  **package** clientpack;  **import** devpack.\*;  **public** **class** ShapeDemo {  **public** **static** **void** main(String[] args) {  Shape s[] = **new** Shape[3];  s[0] = **new** Triangle();  s[1] = **new** Rectangle();  s[2] = **new** Circle();    **for**(**int** i=0; i<s.length; i++) {  s[i].draw();  }  }  } |
| drawing triangle  drawing rectangle  drawing circle |

1. Code this:

public interface PizzaIngredientFactory

{

public Dough createDough();

public Sauce createSauce();

public Cheese createCheese();

public Pepperoni createPepperoni();

public Clams createClams();

}

define 2 implementations

USPizzaIngredientFactory and IndianPizzaIngredientFactory

create an array of "PizzaIngredientFactory" , store all the implemenations object inside it, traverse the array and invoke all the methods.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **public** **interface** PizzaIngredientFactory  {  **public** Dough createDough();  **public** Sauce createSauce();  **public** Cheese createCheese();  **public** Pepperoni createPepperoni();  **public** Clams createClams();  } | | **public** **class** Dough {  **private** String dough;  **public** Dough(String dough) {  **this**.dough = dough;  }    @Override  **public** String toString() {  // **TODO** Auto-generated method stub  **return** "["+dough+"]";  }  } | | **public** **class** Sauce {  **private** String sauce;    **public** Sauce(String sauce){  **this**.sauce=sauce;  }    @Override  **public** String toString() {  // **TODO** Auto-generated method stub  **return** "["+sauce+"]";  }  } | | **public** **class** Cheese {  **private** String cheese;    **public** Cheese(String cheese){  **this**.cheese=cheese;  }    @Override  **public** String toString() {  // **TODO** Auto-generated method stub  **return** "["+cheese+"]";  }  } | | **public** **class** Pepperoni {  **private** String pepperoni;  **public** Pepperoni(String pepperoni) {  **this**.pepperoni = pepperoni;  }    @Override  **public** String toString() {  // **TODO** Auto-generated method stub  **return** "["+pepperoni+"]";  }  } | | **public** **class** Clams {  **private** String clams;  **public** Clams(String clams) {  **this**.clams = clams;  }    @Override  **public** String toString() {  // **TODO** Auto-generated method stub  **return** "["+clams+"]";  }  } | | **public** **class** USPizzaIngredientFactory **implements** PizzaIngredientFactory {  **private** String dough = "all purpose flour";  **private** String sauce = "tomato sauce";  **private** String cheese = "parmessan";  **private** String pepperoni = "pork";  **private** String clams = "white clams";  **public** Dough createDough() {  // **TODO** Auto-generated method stub  **return** **new** Dough(dough);  }  **public** Sauce createSauce() {  // **TODO** Auto-generated method stub  **return** **new** Sauce(sauce);  }  **public** Cheese createCheese() {  // **TODO** Auto-generated method stub  **return** **new** Cheese(cheese);  }  **public** Pepperoni createPepperoni() {  // **TODO** Auto-generated method stub  **return** **new** Pepperoni(pepperoni);  }  **public** Clams createClams() {  // **TODO** Auto-generated method stub  **return** **new** Clams(clams);  }  } | | **public** **class** IndianPizzaIngredientFactory **implements** PizzaIngredientFactory {  **private** String dough = "maida";  **private** String sauce = "tomato and pizza spread";  **private** String cheese = "mozrella";  **private** String pepperoni = "chicken";  **private** String clams = "clams";  **public** Dough createDough() {  // **TODO** Auto-generated method stub  **return** **new** Dough(dough);  }  **public** Sauce createSauce() {  // **TODO** Auto-generated method stub  **return** **new** Sauce(sauce);  }  **public** Cheese createCheese() {  // **TODO** Auto-generated method stub  **return** **new** Cheese(cheese);  }  **public** Pepperoni createPepperoni() {  // **TODO** Auto-generated method stub  **return** **new** Pepperoni(pepperoni);  }  **public** Clams createClams() {  // **TODO** Auto-generated method stub  **return** **new** Clams(clams);  }  } | | **public** **class** PizzaDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  PizzaIngredientFactory pif[] = **new** PizzaIngredientFactory[2];    pif[0] = **new** USPizzaIngredientFactory();  pif[1] = **new** IndianPizzaIngredientFactory();    **for**(**int** i=0; i<pif.length; i++) {  System.***out***.println(pif[i].getClass().getName());  System.***out***.println("\t"+pif[i].createDough());  System.***out***.println("\t"+pif[i].createSauce());  System.***out***.println("\t"+pif[i].createCheese());  System.***out***.println("\t"+pif[i].createPepperoni());  System.***out***.println("\t"+pif[i].createClams());  System.***out***.println();  }  }  } | |
|  |
| USPizzaIngredientFactory  [all purpose flour]  [tomato sauce]  [parmessan]  [pork]  [white clams]  IndianPizzaIngredientFactory  [maida]  [tomato and pizza spread]  [mozrella]  [chicken]  [clams] |

1. Code this:

public class Course

{

public void start()

{

}

public void stop()

{

}

}

now define 3 child classes

DACDBDACourse,MSCit and Basic

inside "DACDBDACourse" class define one more method "public void orientation ()"

define one more class "CourseDemo" with main function.

inside main function

create an array of "Course" of 3 elements.

store all the child classes objects. Traverse the array and invoke "start()" and "stop()" of all the objects. Also invoke "orientation()" method wherever object of "DACDBDACourse" is present.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **public** **class** Course {  **public** **void** start()  {  System.***out***.println("course started");  }  **public** **void** stop()  {  System.***out***.println("course stopped");  }  } | | **public** **class** DACDBDACourse **extends** Course {  **public** **void** orientation () {  System.***out***.println("orientation");  }  } | | **public** **class** MSCit **extends** Course {  } | | **public** **class** Basic **extends** Course {  } | | **public** **class** CourseDemo {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Course c[] = **new** Course[3];  c[0] = **new** DACDBDACourse();  c[1] = **new** MSCit();  c[2] = **new** Basic();    **for**(**int** i=0; i<c.length; i++) {  System.***out***.println(c[i].getClass().getName());  System.***out***.print("\t");  c[i].start();  System.***out***.print("\t");    **if**(c[i] **instanceof** DACDBDACourse) {  DACDBDACourse d = (DACDBDACourse) c[i];  d.orientation();  }  System.***out***.print("\t");  c[i].stop();    System.***out***.println();  }  }  } | |
|  |
| DACDBDACourse  course started  orientation  course stopped  MSCit  course started  course stopped  Basic  course started  course stopped |

1. Code this:

Interface Course

{

void start();

void stop();

}

now define 3 implementations:

DACDBDACourse,MSCit and Basic

inside "DACDBDACourse" class define one more method "public void orientation()"

define one more class "CourseDemo" with main function and "public static void show()" method.

from main function

invoke "show()" method by passing object of any implementations and invoke "start()" and "stop()" methods. Also invoke "orientation()" method whenever object of "DACDBDACourse" is passed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **public** **interface** Course {  **public** **void** start();  **public** **void** stop();  } | | **public** **class** DACDBDACourse **implements** Course {  **public** **void** start()  {  System.***out***.println("course started");  }    **public** **void** stop()  {  System.***out***.println("course stopped");  }    **public** **void** orientation () {  System.***out***.println("orientation");  }  } | | **public** **class** MSCit **implements** Course {  **public** **void** start()  {  System.***out***.println("course started");  }    **public** **void** stop()  {  System.***out***.println("course stopped");  }  } | | **public** **class** Basic **implements** Course {  **public** **void** start()  {  System.***out***.println("course started");  }    **public** **void** stop()  {  System.***out***.println("course stopped");  }  } | | **public** **class** CourseDemo {    **static** **void** show(Course ref) {  System.***out***.println(ref.getClass().getName());  System.***out***.print("\t");  ref.start();  System.***out***.print("\t");    **if**(ref **instanceof** DACDBDACourse) {  DACDBDACourse d = (DACDBDACourse) ref;  d.orientation();  }  System.***out***.print("\t");  ref.stop();    System.***out***.println();  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  *show*(**new** DACDBDACourse());  *show*(**new** MSCit());  *show*(**new** Basic());  }  } | |
|  |
| DACDBDACourse  course started  orientation  course stopped  MSCit  course started  course stopped  Basic  course started  course stopped |

1. Code this:

create a class "Demo" with "disp()" which can take variable number of int arguments and display the sum of all numbers passed to it. invoke "disp()" from main function.

|  |
| --- |
| **public** **class** Demo {    **static** **void** disp(**int** ...num) {  **int** sum=0;    **for**(**int** i=0; i<num.length; i++)  sum+=num[i];    **for**(**int** i=0; i<num.length; i++) {  System.***out***.print(num[i]);  **if**(i!=num.length-1)  System.***out***.print("+");  }  System.***out***.println(" = "+sum);  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  *disp*(10);  *disp*(10,20);  *disp*(10,20,30);  *disp*(10,20,30,40);  *disp*(10,20,30,40,50);  }  } |
|  |
| 10 = 10  10+20 = 30  10+20+30 = 60  10+20+30+40 = 100  10+20+30+40+50 = 150 |