//Employee Hr Admin SalesManager Polymorphic behaviour

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <cstdio>

#include <iostream>

#include <iomanip>

using namespace std;

struct Employee

{

    int id;

    char name[20];

    double salary;

    // default

    Employee()

    {

        this->id = 0;

        strcpy(this->name, "NotGiven");

        this->salary = 0;

    }

    // parameterised Constructor

    Employee(int id, const char \*name, double salary)

    {

        this->id = id;

        strcpy(this->name, name);

        this->salary = salary;

    }

    // setters

    void setId(int id)

    {

        this->id = id;

    }

    void setName(const char \*name)

    {

        strcpy(this->name, name);

    }

    void setSalary(double salary)

    {

        this->salary = salary;

    }

    // gettters

    int getId()

    {

        return this->id;

    }

    char \*getName()

    {

        return this->name;

    }

    double getSalary()

    {

        return this->salary;

    }

    virtual double calculateSalary()

    {

        return this->salary;

    }

    virtual void display()

    {

        cout << "\nId          = " << this->id;

        cout << "\nName        = " << this->name;

        cout << "\nSalary      = " << this->salary;

    }

};

struct SalesManager : public Employee

{

    double Incentive;

    int target;

    // setter

    void setIncentive(double incentive)

    {

        this->Incentive = incentive;

    }

    void setTarget(int target)

    {

        this->target = target;

    }

    // getters

    double getIncentive()

    {

        return this->Incentive;

    }

    int getTarget()

    {

        return this->target;

    }

    // default

    SalesManager() : Employee()

    {

        this->Incentive = 0;

        this->target = 0;

    }

    // parameterised

    SalesManager(int id,const char \*name, double salary, double incentive, int target) : Employee(id, name, salary)

    {

        this->Incentive = incentive;

        this->target = target;

    }

    double calculateSalary()

    {

        return this->Incentive + this->getSalary();

    }

    void display()

    {

        Employee::display();

        cout << "\nIncentive   = " << this->Incentive;

        cout << "\nTarget      = " << this->target;

    }

};

struct Admin : public Employee

{

    double allowance;

    void setAllowance(double allowance)

    {

        this->allowance = allowance;

    }

    double getAllowance()

    {

        return this->allowance;

    }

    Admin() : Employee()

    {

        this->allowance = 0;

    }

    Admin(int id,const char \*name, double salary, double allowance) : Employee(id, name, salary)

    {

        this->allowance = allowance;

    }

    double calculateSalary()

    {

        return this->allowance + this->getSalary();

    }

    void display()

    {

        Employee::display();

        cout << "\nAllowance = " << this->allowance;

    }

};

struct HR : public Employee

{

    double commision;

    void setCommision(double commision)

    {

        this->commision = commision;

    }

    double getCommision()

    {

        return this->commision;

    }

    HR() : Employee()

    {

        this->commision = 0;

    }

    HR(int id,const char \*name, double salary, double commision) : Employee(id, name, salary)

    {

        this->commision = commision;

    }

    double calculateSalary()

    {

        return this->commision + this->getSalary();

    }

    void display()

    {

        Employee::display();

        cout << "\nCommision   = " << this->commision;

    }

};

int main()

{

   Employee \*emp;

   Employee emp1(101,"Suraj",60000);

   emp=&emp1;

   emp->display();

   cout<<"\nTotal Salary = "<<emp->calculateSalary();

   cout<<"\n";

   Admin ad1(102,"Vaishali",45000,2500);

   emp=&ad1;

   emp->display();

   cout<<"\nTotal Salary = "<<emp->calculateSalary();

   cout<<"\n";

   SalesManager sm1(103,"Kale",40000,1200,120);

   emp=&sm1;

   emp->display();

   cout<<"\nTotal Salary = "<<emp->calculateSalary();

   cout<<"\n";

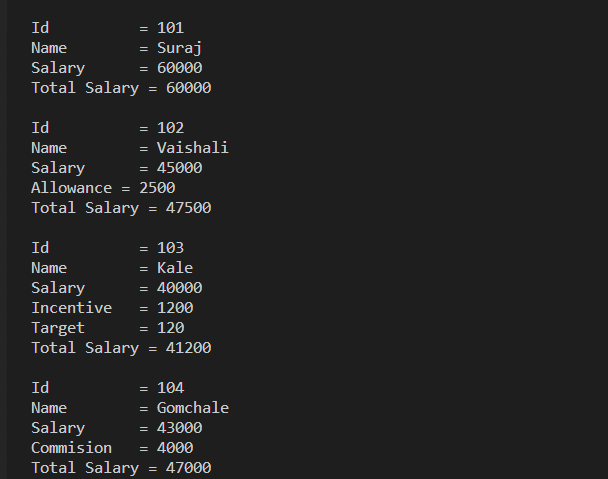
   HR hr1(104,"Gomchale",43000,4000);

   emp=&hr1;

   emp->display();

   cout<<"\nTotal Salary = "<<emp->calculateSalary();

}



#include <iostream>

using namespace std;

float PI = 3.147;

struct Shape

{

    float area;

    Shape()

    {

        this->area=0;

    }

    Shape(float area)

    {

        this->area=area;

    }

    virtual float calculateArea()

    {

        return this->area;

    }

    virtual void display()

    {

        cout<<"\nArea = "<<area;

    }

};

struct circle : public Shape

{

    float r;

    void setRadius(float r)

    {

        this->r = r;

    }

    float getRadius()

    {

        return this->r;

    }

    circle()

    {

        this->r = 0;

    }

    circle(float r)

    {

        this->r = r;

    }

    void display()

    {

        cout << "\nradius  = " << this->r;

    }

    float calculateArea()

    {

        return this->area = PI \* this->r \* this->r;

    }

};

struct tringle : public Shape

{

    float b, h;

    void setBredth(float b)

    {

        this->b = b;

    }

    void setHeight(float h)

    {

        this->h = h;

    }

    float getBredth()

    {

        return this->b;

    }

    float getHeigth()

    {

        return this->h;

    }

    tringle()

    {

        this->b = 0;

        this->h = 0;

    }

    tringle(float b, float h)

    {

        this->b = b;

        this->h = h;

    }

    void display()

    {

        cout << "\nBredth  = " << this->b;

        cout << "\nHeigth  = " << this->h;

    }

    float calculateArea()

    {

        return this->b \* this->h;

    }

};

struct rectangle : public Shape

{

    float l, w;

    rectangle()

    {

        this->l = 0;

        this->w = 0;

    }

    rectangle(float l, float w)

    {

        this->l = l;

        this->w = w;

    }

    void setLength(float l)

    {

        this->l = l;

    }

    void setWidth(float w)

    {

        this->w = w;

    }

    float getLength()

    {

        return this->l;

    }

    float getWidth()

    {

        return this->w;

    }

    void display()

    {

        cout << "\nLenth  = " << l;

        cout << "\nWidth  = " << w;

    }

    float calculateArea()

    {

        area = 2 \* l \* w;

        return area;

    }

};

int main()

{

    Shape \*sh;

    circle c1(56);

    sh=&c1;

    sh->display();

    cout<<"\nArea = "<<sh->calculateArea();

    rectangle r1(10, 20);

    sh=&r1;

    cout<<"\n";

    sh->display();

    cout<<"\nArea = "<<sh->calculateArea();

    tringle t1(30, 40);

    sh=&t1;

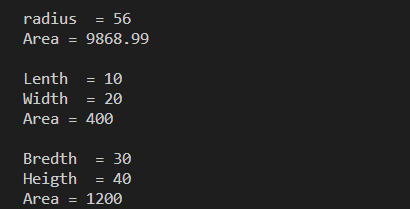
    cout<<"\n";

    sh->display();

    cout<<"\nArea = "<<sh->calculateArea();

}

OUTPUT -



// 3. Write a code to implement inheritance where vehicle is base class and derived

// classes like bike, car, bus etc.

#include <iostream>

using namespace std;

struct vehicle

{

    int noOfWheels;

    vehicle()

    {

        this->noOfWheels=0;

    }

    vehicle(int wheel)

    {

        this->noOfWheels=wheel;

    }

    void setWheels(int wheel)

    {

        this->noOfWheels = wheel;

    }

    int getWheels()

    {

        return this->noOfWheels;

    }

    virtual void display()

    {

        cout << "\nNo of wheels = " << this->noOfWheels;

    }

};

struct bus : public vehicle

{

    int noofWindow;

    bus():vehicle()

    {

        this->noofWindow=0;

    }

    bus(int wheel, int window):vehicle(wheel)

    {

        this->noofWindow=window;

    }

    void setWindows(int window)

    {

        this->noofWindow=window;

    }

    int getWindows()

    {

        return this->noofWindow;

    }

    void display()

    {

        vehicle::display();

        cout<<"\nNo of Windows = "<<this->noofWindow;

    }

};

struct car : public vehicle

{

    int noofWindow;

    car():vehicle()

    {

        this->noofWindow=0;

    }

    car(int wheel, int window):vehicle(wheel)

    {

        this->noofWindow=window;

    }

    void setWindows(int window)

    {

        this->noofWindow=window;

    }

    int getWindows()

    {

        return this->noofWindow;

    }

    void display()

    {

        vehicle::display();

        cout<<"\nNo of Windows = "<<this->noofWindow;

    }

};

struct bike : public vehicle

{

    int noofShockups;

    bike():vehicle()

    {

        this->noofShockups=0;

    }

    bike(int wheel,int shockup):vehicle(wheel)

    {

        this->noofShockups=shockup;

    }

    void setShockup(int shockup)

    {

        this->noofShockups=shockup;

    }

    int getShockup()

    {

        return this->noofShockups;

    }

    void display()

    {

        vehicle::display();

        cout<<"\nNo of Shockups = "<<this->noofShockups;

    }

};

int main()

{

    vehicle \* v;

    bus b1(10,20);

    v=&b1;

    cout<<"\n\nBike Info";

    v->display();

    car c1(4,4);

    v=&c1;

    cout<<"\n\nCar Info";

    v->display();

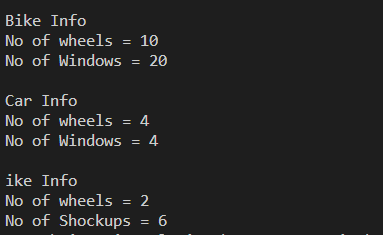
    bike bk1(2,6);

    v=&bk1;

    cout<<"\n\nike Info";

    v->display();

}



//mouse heirarchy

#include<iostream>

#include<string.h>

using namespace std;

class mouse

{

    protected:

    int productId;

    public:

    mouse()

    {

        this->productId=0;

    }

    mouse(int id)

    {

        this->productId=id;

    }

    void setProductId(int id)

    {

        this->productId=id;

    }

    int getProductId()

    {

        return this->productId;

    }

    virtual void display()

    {

        cout<<"\nProductId = "<<this->productId;

    }

};

class opticalMouse:public mouse

{

    protected:

    const char \*sensorType;

    public:

    opticalMouse():mouse()

    {

        // strcpy(this->sensorType,"NotGiven");

        this->sensorType="NotGiven";

    }

    opticalMouse(int id,const char\* sensorType):mouse(id)

    {

        // strcpy(this->sensorType,sensorType);

        this->sensorType=sensorType;

    }

    void setSensorType(const char\* sensorType)

    {

        // strcpy(this->sensorType,sensorType);

        this->sensorType=sensorType;

    }

    const char\* getSensorType()

    {

        return sensorType;

    }

    void display()

    {

        mouse::display();

        cout<<"\nSensorType = "<<this->sensorType;

    }

};

class ballMouse:public mouse

{

    protected:

    const char \*ballType;

    public:

    ballMouse():mouse()

    {

        // strcpy(this->ballType,"NotGiven");

        this->ballType="NotGiven";

    }

    ballMouse(int id,const char\* ballType):mouse(id)

    {

        // strcpy(this->ballType,ballType);

        this->ballType=ballType;

    }

    void setBallType(const char\* ballType)

    {

        // strcpy(this->ballType,ballType);

        this->ballType=ballType;

    }

    const char\* getBallType()

    {

        return this->ballType;

    }

    void display()

    {

        mouse::display();

        cout<<"\nBallType = "<<this->ballType;

    }

};

int main()

{

    const char \*sensor="Laser";

    const char\* ballType="Rubber";

    mouse \*mouse;

    opticalMouse op1(101,sensor);

    // op1.setProductId(101);

    // op1.setSensorType(sensor);

    mouse=&op1;

    mouse->display();

    ballMouse b1(102,ballType);

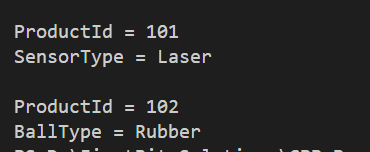
    cout<<"\n";

    mouse=&b1;

    mouse->display();

    return 0;

}



// artist painter musician

#include <iostream>

#include <string.h>

using namespace std;

class artist

{

protected:

    const char \*name;

    int age;

    const char \*gender;

public:

    // Default constructor suru

    artist()

    {

        // strcpy(this->name, "NotGiven");

        this->name="NotGiven";

        this->age = 0;

        // strcpy(this->gender, "NotDefine");

        this->gender="NotDefine";

    }

    // Parameterised constructor suru

    artist(const char \*name, int age,const  char \*gender)

    {

        // strcpy(this->name, name);

        this->name=name;

        this->age = age;

        // strcpy(this->gender, gender);

        this->gender=gender;

    }

    // setters

    void setName(const char \*name)

    {

        // strcpy(this->name, name);

        this->name=name;

    }

    void setAge(int age)

    {

        this->age = age;

    }

    void setGender(const char \*gender)

    {

        // strcpy(this->gender, gender);

        this->gender=gender;

    }

    // getters

    const char \*getName()

    {

        return this->name;

    }

    int getAge()

    {

        return this->age;

    }

    const char \*getGender()

    {

        return this->gender;

    }

    // Display

    void display()

    {

        cout << "\nName   = " << this->name;

        cout << "\nAge    = " << this->age;

        cout << "\nGender = " << this->gender;

    }

};

class Painter : public artist

{

protected:

    int noOfBrush;

    char paintingType[20];

public:

    Painter() : artist()

    {

        this->noOfBrush = 0;

        strcpy(this->paintingType, "NotSpecified");

        // this->paintingType=

    }

    Painter(const char \*name, int age,const  char \*gender, int noOfBrush, const char \*paintingType) : artist(name, age, gender)

    {

        this->noOfBrush = noOfBrush;

        strcpy(this->paintingType, paintingType);

        // this->paintingType=paintingType;

    }

    void setBrush(int noOfBrush)

    {

        this->noOfBrush = noOfBrush;

    }

    void setPaintingType(const char \*paintingType)

    {

        strcpy(this->paintingType, paintingType);

        // this->paintingType=paintingType;

    }

    int getBrush()

    {

        return this->noOfBrush;

    }

    char \*getPaintingType()

    {

        return this->paintingType;

    }

    void display()

    {

        artist::display();

        cout << "\nNoOfBrush    = " << this->noOfBrush;

        cout << "\nPaintingType = " << this->paintingType;

    }

};

int main()

{

    artist a1;

    const char name[10] = "Vivek";

    const char gender[10] = "Male";

    const char name1[10] = "Shubham";

    const char paintingType[20] = "OilPainting";

    Painter p1(name, 81, gender, 5, paintingType);

    p1.display();

}

