// Demonstrate class inheritance and polymorphism
// George F. Riley, Georgia Tech, Spring 2004

#include <iostream>
#include <math.h>
using namespace std;

class Car {
public:
    Car() : position(0), velocity(0), acceleration(0) {}
    void Print();
    // All cars must be able to accelerate, but different cars do it differently
    virtual void Accelerate(double a) = 0;
    void UpdatePosition(double now);
    bool Stopped();
private:
    double position;
    double velocity;
protected:
    double acceleration;
};

void Car::Print()
{
    cout << "Position is " << position
        << " velocity is " << velocity
        << " acceleration is " << acceleration << endl;
}

void Car::UpdatePosition(double elapsed)
{
    double initVelocity = velocity;
    // New velocity is is old velocity + acceleration*elapsed time
    velocity += acceleration*elapsed;
    // New position assumes average
    position += (initVelocity + (velocity-initVelocity)/2.0)*elapsed;
}

bool Car::Stopped()
{ // True if car is stopped
    return (velocity == 0.0);
}

// Now define a Yugo as subclass of Car
class Yugo : public Car {
public:
    // All Cars must define the accelerate procedure
    void Accelerate(double a);
};

void Yugo::Accelerate(double a)
{ // Yugo’s can’t accelerate more then 0.2 meters per second * second
    if (abs(a) > 0.2) a = 0.2 * a / abs(a);
    acceleration = a;
}
// Now define a Ferrari subclass
class Ferrari : public Car
{
public:
    void Accelerate(double s);
};

void Ferrari::Accelerate(double a)
{
    // Ferrari’s can’t accelerate more than 10 meters per second * second
    if (abs(a) > 10.0) a = 10.0 * a / abs(a);
    acceleration = a;
}

void AdvanceTime(Car& c, double elapsed)
{
    // Advances time for a Car object
    c.UpdatePosition(elapsed);
    c.Print();
}

int main()
{
    // Car c; // Won’t compile, can’t create cars
    Yugo y; // A Yugo
    Ferrari f; // A Ferrari

    y.Accelerate(5);
    f.Accelerate(5);
    AdvanceTime(y, 40);
    AdvanceTime(f, 40);
}

Program inheritance.cc (continued)