CSE 231 Problem Set 06

Each problem in the problem set will start with the following two classes:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* VELOCITY

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Velocity

{

public:

Velocity() : dx(0.0), dy(0.0) {}

Velocity(double dx, double dy) : dx(dx), dy(dy) {}

double getDx() const { return dx; }

double getDy() const { return dy; }

void setDx(double dx) { this->dx = dx; }

void setDy(double dy) { this->dy = dy; }

private:

double dx;

double dy;

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* POSITION

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class Position

{

public:

Position() : x(0.0), y(0.0) {}

Position(double x, double y) : x(x), y(y) {}

double getX() const { return x; }

double getY() const { return y; }

void setX(double x) { this->x = x; }

void setY(double y) { this->y = y; }

Position& add(const Velocity& v, double t)

{

x += v.getDx() \* t;

y += v.getDy() \* t;

return \*this;

}

Position& addMany(const Velocity& v, double t, int count)

{

for (int i = 0; i < count; i++)

add(v, t);

return \*this;

}

private:

double x;

double y;

};

Please put this code in an editor, add a main(), and make sure it runs.

# Problem 34.1: Dummy

Create a dummy double for the Velocity class.

class Velocity

{

public:

Velocity() {};

double getDx() const { assert(false); }

double getDy() const { assert(false); }

void setDx(double dx) { assert(false); }

void setDy(double dy) { assert(false); }

};

# Problem 34.2: Stub

Create two stubs for the Velocity class: One that returns DX,DY of 1.1,2.2, and a second that returns a DX,DY of 11.1, 22.2.

class Velocity

{

public:

Velocity() {};

double getDx() const { return 1.1; }

double getDy() const { return 2.2; }

void setDx(double dx) { assert(false); }

void setDy(double dy) { assert(false); }

};

class Velocity

{

public:

Velocity() {};

double getDx() const { return 11.1; }

double getDy() const { return 22.2; }

void setDx(double dx) { assert(false); }

void setDy(double dy) { assert(false); }

};

# Problem 34.3: Stub Utilization

There is the following unit test for the Position class:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TEST POSITION

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class TestPosition

{

public:

static void run()

{

// Position::add()

add\_small();

add\_large();

}

private:

//

// Position::add

//

// Add a small velocity to a position

static void add\_small()

{ // SETUP

Position p;

p.x = 9.4;

p.y = 7.3;

Velocity v(1.1, 2.2);

double t = 1.0;

// EXERCISE

p.add(v, t);

// VERIFY

assert(p.x == 10.5);

assert(p.y == 9.5);

assert(v.getDx() == 1.1);

assert(v.getDy() == 2.2);

assert(t == 1.0);

} // TEARDOWN

// Add a large velocity to a position

static void add\_large()

{ // SETUP

Position p;

p.x = 9.4;

p.y = 7.3;

Velocity v(11.1, 22.2);

double t = 1.0;

// EXERCISE

p.add(v, t);

// VERIFY

assert(p.x == 20.5);

assert(p.y == 29.5);

assert(v.getDx() == 11.1);

assert(v.getDy() == 22.2);

assert(t == 1.0);

} // TEARDOWN

};

Please copy this into your code and run it from main:

int main()

{

TestPosition::run();

cout << "Tests pass\n";

}

Modify the above code to utilize the stubs created in 34.2.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* TEST POSITION

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class TestPosition

{

public:

static void run()

{

// Position::add()

add\_small();

add\_large();

}

private:

//

// Position::add

[ //

// Add a small velocity to a position

static void add\_small()

{ // SETUP

Position\_1 p;

p.x = 9.4;

p.y = 7.3;

Velocity\_1 v(1.1, 2.2);

double t = 1.0;

// EXERCISE

p.add(v, t);

// VERIFY

assert(p.x == 10.5);

assert(p.y == 9.5);

assert(v.getDx() == 1.1);

assert(v.getDy() == 2.2);

assert(t == 1.0);

} // TEARDOWN

// Add a large velocity to a position

static void add\_large()

{ // SETUP

Position\_2 p;

p.x = 9.4;

p.y = 7.3;

Velocity\_2 v(11.1, 22.2);

double t = 1.0;

// EXERCISE

p.add(v, t);

// VERIFY

assert(p.x == 20.5);

assert(p.y == 29.5);

assert(v.getDx() == 11.1);

assert(v.getDy() == 22.2);

assert(t == 1.0); ]

} // TEARDOWN

};

# Problem 34.4: Spy

Create a spy double for the Velocity class. This double will count the number of times that Velocity::getDx() or Velocity::getDy() are called. Note that this spy double will be a stub that always returns DX,DY = 1.1, 2.2.

class spy

{

public:

int DxTracker = 0;

int DyTracker = 0;

int getDxTracker() { return DxTracker; }

int getDYTracker() { return DyTracker; }

};

class Velocity : public spy

{

public:

Velocity() {};

Velocity(double dx, double dy) : dx(dx), dy(dy) {}

double getDx()

{

DxTracker++;

return 1.1;

}

double getDy()

{

DyTracker++;

return 2.2;

}

void setDx(double dx) { assert(false); }

void setDy(double dy) { assert(false); }

private:

double dx;

double dy;

};

# Problem 34.5: Spy Utilization

Add the following unit test to the TestPosition class:

//

// Position::addMany

//

// Add acceleration five times

static void addMany()

{ // SETUP

Position p;

p.x = 10.0;

p.y = 20.0;

Velocity v(1.1, 2.2);

double t = 1.0;

int n = 3;

// EXERCISE

p.addMany(v, t, n);

// VERIFY

assert(13.299 <= p.x && p.x <= 13.301);

assert(26.599 <= p.y && p.y <= 26.601);

assert(t == 1.0);

assert(n == 3);

assert(v.getDx() == 1.1);

assert(v.getDy() == 2.2);

} // TEARDOWN

Note that you will also have to call this method from run(). Please modify the above method to include your spy form 34.4. Note that you should add an assert to ensure the expected number of calls to getDx() and getDy() were performed.