**Logitech**

HW07

1. Abstract Factory

TEST

{

CartPartFactory::get\_instance()-> register\_factory( WoodenFactory::get\_instance());

CartPartFactory::get\_instance()-> register\_factory( SteelFactory::get\_instance());

CarPart\* wooden\_wheel = CartPartFactory::get\_instance()->create\_wheel(“wooden”);

CarPart\* steel\_wheel = CartPartFactory::get\_instance()->create\_wheel(“steel”);

CarPart\* wooden\_frame = CartPartFactory::get\_instance()->create\_frame(“wooden”);

CarPart\* steel\_frame = CartPartFactory::get\_instance()->create\_frame(“steel”);

EXPECT\_STREQ( “CarPart: wooden wheel”, wooden\_wheel.get\_info() );

EXPECT\_STREQ( “CarPart: wooden frame”, wooden\_frame.get\_info() );

EXPECT\_STREQ( “CarPart: steel wheel”, steel\_wheel.get\_info() );

EXPECT\_STREQ( “CarPart: steel frame”, steel\_frame.get\_info() );

}

1. Composite

TEST

{

Employee\* A = new Worker (“Mary”);

Employee\* B = new Worker (“Jean”);

Employee\* C = new Manager (“John Doe”);

Employee\* D = new Director (“Gin”);

C.manage(A);

C.manage(B);

D.manage(C);

EXPECT\_STREQ( “Gin John Doe Mary Jean”, D->who() );

//delete...

}

1. Prototype

TEST

{

std::vector<Image\*> all\_image\_prototype;

all\_images.push\_back ( new DogImage(“big”) );

all\_images.push\_back ( new CatImage(“small”) );

all\_images.push\_back ( new DogImage(“huge”) );

Image\* my\_image = all\_images[1].clone();

EXPECT\_STREQ( “cat image: small”, my\_image->info() );

my\_image = all\_images[2].clone();

EXPECT\_STREQ( “dog image: huge”, my\_image->info() );

//delete...

}