**public** **class** CEO {

**private** **static** CEO *ceo*=**new** CEO();

**private** CEO(){

System.***out***.println("CEO()");

}

**public** **static** CEO getInstance(){

System.***out***.println("getInstance");

**if**(*ceo*==**null**){

*ceo*=**new** CEO();

}

**return** *ceo*;

}

**public** String getCEOName(String name){

System.***out***.println("getCEOName");

**return** name;

}

}

for debugging purpose I have added some println stmt inside each block

this is our Singleton class

so situation can be different whether we want to mock the whole class or mock some method of the class or mock somefields of the class which are used in our test method

mock the whole singleton class:-

**import** **static** org.junit.Assert.\*;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.mockito.Mock;

**import** **static** org.powermock.api.mockito.PowerMockito.\*;

**import** org.powermock.core.classloader.annotations.PrepareForTest;

**import** org.powermock.modules.junit4.PowerMockRunner;

@RunWith(PowerMockRunner.**class**)

@PrepareForTest({CEO.**class**})

**public** **class** CEOTest {

@Mock

CEO ceo;

@Test

**public** **void** test() {

*mockStatic*(CEO.**class**);

*when*(CEO.*getInstance*()).thenReturn(ceo);

*when*(ceo.getCEOName("Deepak")).thenReturn("David");

System.***out***.println(ceo.getCEOName("Deepak"));

System.***out***.println(ceo.getCEOName("ABC"));

}

}

o/p

CEO()

David

null

**import** **static** org.junit.Assert.\*;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.mockito.Mock;

**import** **static** org.powermock.api.mockito.PowerMockito.\*;

**import** org.powermock.core.classloader.annotations.PrepareForTest;

**import** org.powermock.modules.junit4.PowerMockRunner;

@RunWith(PowerMockRunner.**class**)

@PrepareForTest({CEO.**class**})

**public** **class** CEOTest {

@Mock

CEO ceo;

@Test

**public** **void** test() {

*suppress*(*constructor*(CEO.**class**));

*mockStatic*(CEO.**class**);

*when*(CEO.*getInstance*()).thenReturn(ceo);

*when*(ceo.getCEOName("Deepak")).thenReturn("David");

System.***out***.println(ceo.getCEOName("Deepak"));

System.***out***.println(ceo.getCEOName("ABC"));

}

}

here this case differs from the first case only be suppress constructor.Still it has no effect

as the ouput is same

suppose the singleton class is available to you that is yiu can getObjct bt some methods of the class is not available then in that case we can create a partial mock

**public** **class** CEO {

**private** **static** CEO *ceo*=**new** CEO();

**private** CEO(){

System.***out***.println("CEO()");

}

**public** **static** CEO getInstance(){

System.***out***.println("getInstance");

**if**(*ceo*==**null**){

*ceo*=**new** CEO();

}

**return** *ceo*;

}

**public** String getCEOName(String name){

System.***out***.println("getCEOName");

**return** name;

}

**public** String getCEOAddress(String address){

System.***out***.println("getCEOAddress");

**return** address;

}

}

**import** **static** org.junit.Assert.\*;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.mockito.Mock;

**import** **static** org.powermock.api.mockito.PowerMockito.\*;

**import** org.powermock.core.classloader.annotations.PrepareForTest;

**import** org.powermock.modules.junit4.PowerMockRunner;

@RunWith(PowerMockRunner.**class**)

@PrepareForTest({CEO.**class**})

**public** **class** CEOTest {

CEO ceo;

@Test

**public** **void** test() {

ceo=CEO.*getInstance*();

CEO spyCEO=*spy*(ceo);

*when*(spyCEO.getCEOName("ABC")).thenReturn("DEF");

System.***out***.println(spyCEO.getCEOName("ABC"));

System.***out***.println(spyCEO.getCEOAddress("Bangalore"));

}

}

op

CEO()

getInstance

getCEOName

DEF

getCEOAddress

Bangalore