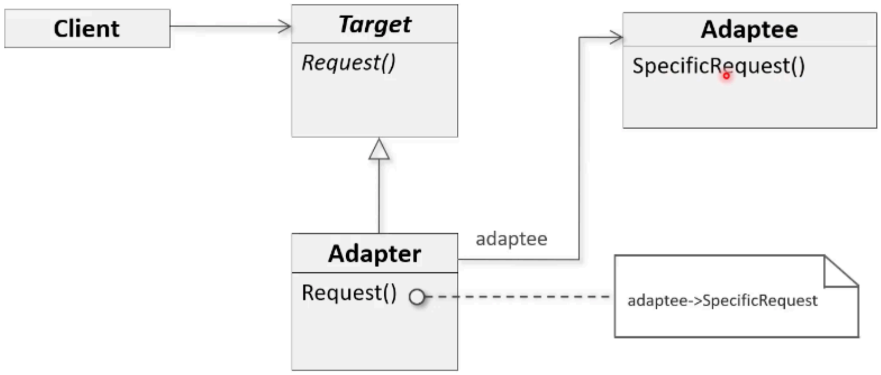
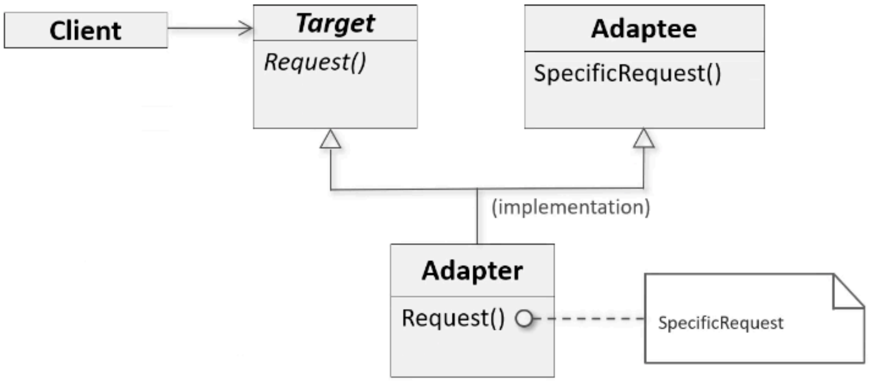
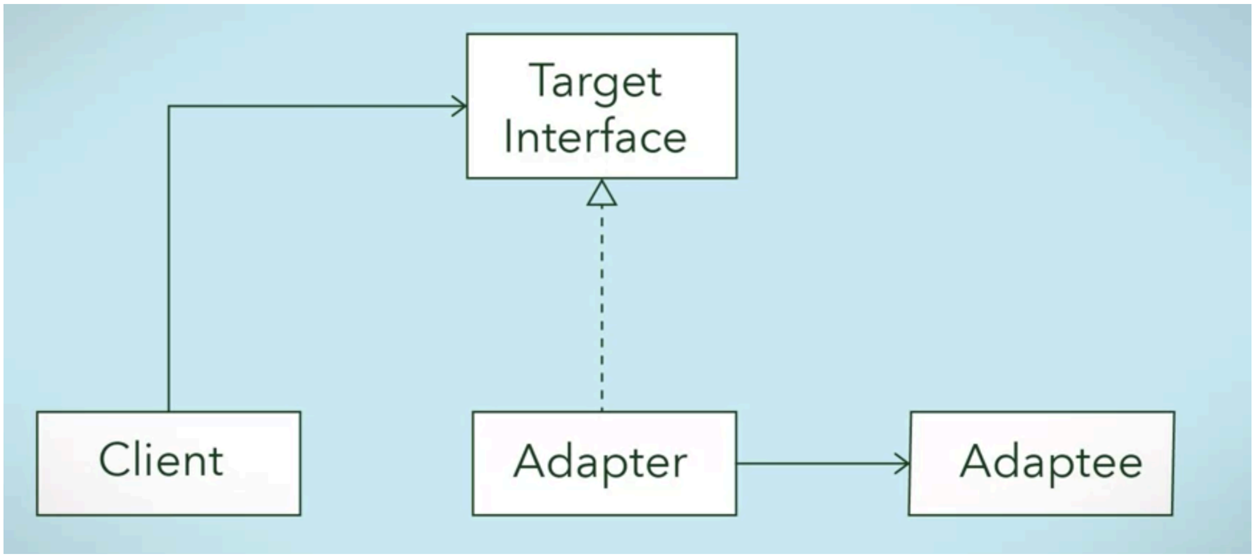
Basic use of Adaptor pattern incompatible class .

Object Adapter



Class Adapter





Question

What type of class is the adapter an example of?

○ Superclass

● Wrapper class

○ Interface

○ Abstract class

✓ Correct

The adapter essentially encapsulates the adaptee and presents a new interface, or appearance, to the client class. It does this by wrapping the adaptee’s interface and exposing a new target interface that makes sense to the client. This is the correct answer.

Skip Continue

What are the characteristics of the adapter design pattern?

☑ The client and adaptee classes have incompatible interfaces.

✓ Correct

☑ An adapter is a wrapper class that wraps the adaptee, hiding it from the client.

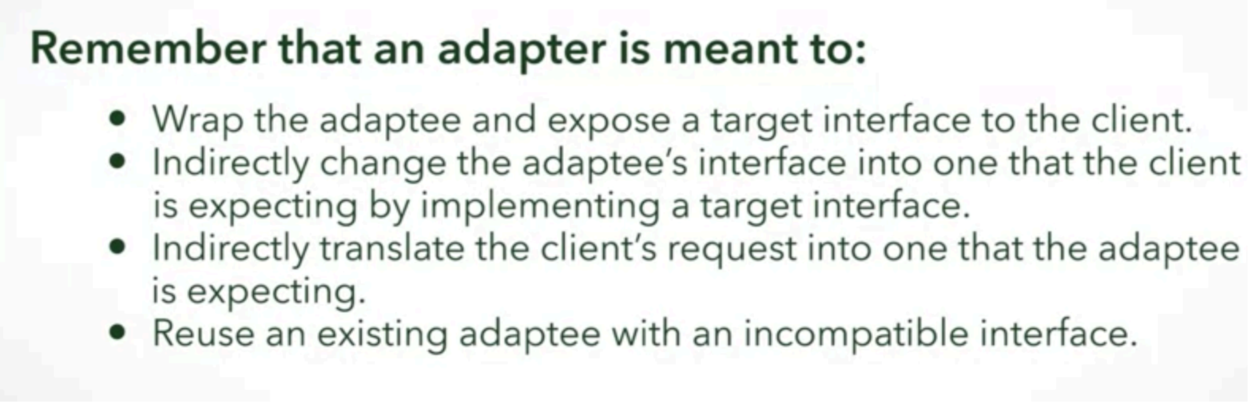
✓ Correct

☑ The client sends requests indirectly to the adaptee by using the adapter’s target interface.

✓ Correct

☑ The adapter translates the request sent by the client class into a request that the adaptee class is expecting.

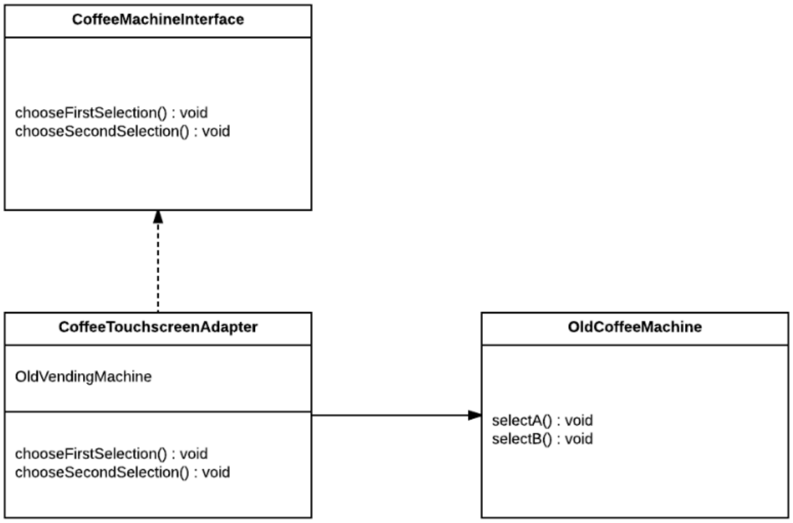
✓ Correct



Remember that an adapter is meant to:

- Wrap the adaptee and expose a target interface to the client.
- Indirectly change the adaptee's interface into one that the client is expecting by implementing a target interface.
- Indirectly translate the client's request into one that the adaptee is expecting.
- Reuse an existing adaptee with an incompatible interface.

```
// Adapter Design Pattern Example Code

#include <iostream>

// Target Interface
class Printer {
public:
    virtual void print() = 0;
};

// Adaptee
class LegacyPrinter {
public:
    void printDocument() {
        std::cout << "Legacy Printer is printing a document." <<
std::endl;
    }
};

// Adapter
class PrinterAdapter : public Printer {
private:
    LegacyPrinter legacyPrinter;

public:
    void print() override {
        legacyPrinter.printDocument();
    }
};

// Client Code
void clientCode(Printer& printer) {
    printer.print();
}

int main() {
    // Using the Adapter
    PrinterAdapter adapter;
    clientCode(adapter);

    return 0;
}
```


CoffeeMachineInterface
chooseFirstSelection() : void
chooseSecondSelection() : void
CoffeeTouchscreenAdapter
OldVendingMachine
chooseFirstSelection() : void
chooseSecondSelection() : void
OldCoffeeMachine
selectA() : void
selectB() : void

```
/******************************************************************************

                              Online C++ Compiler.
               Code, Compile, Run and Debug C++ program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <iostream>
using namespace std;

//adptee
class oldCoffeeMachine {
    public:
    void selectA() {
        cout<<" adptee A selected";
    }

    void selectB() {
        cout<<" adptee B selected";
    }
};

//interface
class ICoffeeMachine{
    public :
    void virtual chooseFirst() =0;
    void virtual chooseSecond() =0;

};

//adaptor
class NewCoffeeMachine : public ICoffeeMachine {
    oldCoffeeMachine oldvending_machine;
    public:
     void  chooseFirst() {
         cout<<"\n Touch screen First selected";
         oldvending_machine.selectA();
     }
    void  chooseSecond() {
        cout<<"\n Touch screen Second selected";
        oldvending_machine.selectB();
    }
};


void clientcode(NewCoffeeMachine &adaptor) {
    adaptor.chooseFirst();
    adaptor.chooseSecond();

}

int main()
{
    std::cout<<"Hello World";

    NewCoffeeMachine touchscreenMachine;
    clientcode(touchscreenMachine);
    return 0;
}
```