**//Parent Class**

Public class PriceCalculator {

//class variables

quoteState;

init(quoteState){

this.quoteState = quoteState;

}

performNewSubCalculations() {}

performAmendSubCalculations() {}

performRenewSubCalculations(){}

async doCalculations() {

if (commons.isAmendSub(this.quoteState)) {

await this.performAmendSubCalculations();

return;

}

if (commons.isRenewSub(this.quoteState)) {

await this.performRenewSubCalculations();

return;

}

await this.performNewSubCalculations();

}

//Common helpers for all extending child classes

isRamped() {}

getRampGroupedCharges() {}

findPSTimelineVersionCharge(timelines) {}

calculateARR(timelines, supportLevelToCheck) {}

calculateChargeARR(charge) {}

calculateMinListPrice(config, periodData) {}

findMatchingPSConfig(serviceCode) {}

calculateNewListPrice(matchingPSConfiguration, totalARR, periodData) {}

}

**//Child Classes**

class DirectPriceCalculator extends PriceCalculator {

async performNewSubCalculations() {

//New Subscription Logic

}

async performAmendSubCalculations() {

//Amend Subscription Logic

}

async performRenewSubCalculations() {

//Renew Subscription Logic

}

}

class IndirectPriceCalculator extends PriceCalculator {

async performNewSubCalculations() {}

async performAmendSubCalculations() {}

async performRenewSubCalculations() {}

}

**//Factory Class**

class PriceCalculatorFactory {

static \_calculators = [

new DirectPriceCalculator(),

new IndirectPriceCalculator(),

];

static async calculate(quoteState, customAction, params) {

for(const calculator of \_calculators) {

calculator.doCalculations();

}

}

}

**//Main Class to invoke factory Class**

export class TSPremierListPriceCalculateAction extends commons.CustomAction {

async perform(params) {

await PriceCalculatorFactory.calculate(this.quoteState, this, params);

}

}