**Premium Calculation Service Implementation Class**

@Override

**public** PremiumCalculationSuccessRes savePremiumCalculation(PremiumCalculationMasterReq req) {

PremiumCalculationSuccessRes res = **new** PremiumCalculationSuccessRes();

//PremiumCalculationMaster entity = new PremiumCalculationMaster();

SimpleDateFormat sdf = **new** SimpleDateFormat("dd/MM/yyyy");

ModelMapper mapper = **new** ModelMapper();

Date entryDate = **null**;

**try**

{

BigDecimal totalIns=BigDecimal.***ZERO***;

BigDecimal premiumAmount = BigDecimal.***ZERO***;

BigDecimal sumInsuredAmount = BigDecimal.***ZERO***;

BigDecimal sumInsuredPercentage =BigDecimal.***ZERO***;

BigDecimal sumInsuredAmountPer = BigDecimal.***ZERO***;

BigDecimal totalPercentage = BigDecimal.***ZERO***;

BigDecimal totalPremium = BigDecimal.***ZERO***;

SimpleDateFormat sdf1 = **new** SimpleDateFormat("dd/MM/yyyy");

res= mapper.map(req, PremiumCalculationSuccessRes.**class**);

BigDecimal sumAmount =**new** BigDecimal(req.getSumInsAmt());

**if**( **new** BigDecimal(0).compareTo( sumAmount) < 0 && sumAmount.compareTo( **new** BigDecimal(100000)) < 0 ) {

sumInsuredPercentage = **new** BigDecimal(10);

}**else** **if**( **new** BigDecimal(100001).compareTo( sumAmount) < 0 && sumAmount.compareTo( **new** BigDecimal(200000)) < 0 ) {

sumInsuredPercentage = **new** BigDecimal(20);

}

**else** **if**( **new** BigDecimal(200001).compareTo( sumAmount) < 0 && sumAmount.compareTo( **new** BigDecimal(500000)) < 0 ) {

sumInsuredPercentage = **new** BigDecimal(30);

}

sumInsuredAmountPer = (sumInsuredAmount.multiply(sumInsuredPercentage)).divide(**new** BigDecimal(100));

//// Local Date Time

/\*

\* LocalDateTime vehAge= sdf.parse(req.getVehAge());

\* Calendar cal = Calendar.getInstance();

\* LocalDateTime today = sdf.parse(cal.getTime().toString());

\*/

//// Local Date

/\*

\* LocalDate vehAge= sdf.parse(req.getVehAge().toString();

\* Calendar cal = Calendar.getInstance();

\* LocalDate today = sdf.format(cal.getTime().toString());

\*

\*/

////Date

Date vehAge= sdf.parse(req.getVehAge());

Calendar cal = Calendar.*getInstance*();

Date today = **new** Date();

**long** years = today.~~getYear~~()-vehAge.~~getYear~~();

BigDecimal vehPercentage = BigDecimal.***ZERO***;

BigDecimal vehInsurance = BigDecimal.***ZERO***;

**if**(years<=1)

{

vehPercentage= **new** BigDecimal(0);

}

**else** **if**(years>1&&years<=5)

{

vehPercentage= **new** BigDecimal(10);

}

**else** **if**(years>=5&&years<=10)

{

vehPercentage= **new** BigDecimal(25);

}

vehInsurance = (sumInsuredAmount.multiply(vehPercentage)).divide(**new** BigDecimal(100));

Date driverAge= sdf.parse(req.getDriverAge());

Calendar cal1 = Calendar.*getInstance*();

Date today2 = **new** Date();

**int** years1 = today2.~~getYear~~()-driverAge.~~getYear~~();

BigDecimal driverPercentage = BigDecimal.***ZERO***;

BigDecimal driverInsurance = BigDecimal.***ZERO***;

**if**(years1>=18 && years1 <=25)

{

driverPercentage= **new** BigDecimal(10);

}

**else** **if**(years1>=26&&years1<=40)

{

driverPercentage= **new** BigDecimal(5);

}

driverInsurance = (sumInsuredAmount.multiply(driverPercentage)).divide(**new** BigDecimal(100));

String bodyType =req.getVehType();

String bodyType1= "saloon";

String bodyType2 = "4wheel";

String bodyType3 = "sports";

BigDecimal bodyPercentage = **new** BigDecimal(0);

BigDecimal bodyInsurance = **new** BigDecimal(0);

**if**(bodyType.equals(bodyType1))

{

bodyPercentage = **new** BigDecimal(1.5);

}

**else** **if**(bodyType.equals(bodyType2))

{

bodyPercentage = **new** BigDecimal(2);

}

**else** **if**(bodyType.equals(bodyType3))

{

bodyPercentage = **new** BigDecimal(3);

}

bodyInsurance = (sumInsuredAmount.multiply(bodyPercentage)).divide(**new** BigDecimal(100));

totalPercentage = sumInsuredAmountPer.add(vehInsurance).add(driverInsurance).add(bodyInsurance);

premiumAmount =sumInsuredAmount.multiply(totalPercentage);

res.setResponse("Saved Successfully ");

res.setPremiumAmount(premiumAmount.toString());

/\*

\* entity.setDriAgePer(Double.valueOf(req.getDriverAge()));

\* entity.setTotIns(Double.valueOf(req.getSumInsAmt()));

\* entity.setTotPerCal(null) entity.setExpiryDate(sdf.parse(expiryDate));

\*/

} **catch** (Exception ex) {

log.error(ex);

**return** **null**;

}

**return** res;

}