package com.checkout.payments.addtip  
  
import android.view.View  
import com.chase.mocks.TestFactory  
import com.chase.payments.CurrencyFormatter  
import com.chase.payments.Payment  
import com.chase.payments.PercentFormatter  
import com.chase.preferences.tip.Tip  
import com.chase.preferences.tip.TipsUseCase  
import com.cp.mpos.R  
import junit.framework.TestCase.assertEquals  
import org.junit.After  
import org.junit.Before  
import org.junit.Test  
import org.junit.runner.RunWith  
import org.mockito.Mock  
import org.mockito.Mockito  
import org.mockito.MockitoAnnotations  
import org.robolectric.RobolectricTestRunner  
import java.math.BigDecimal  
import org.mockito.Mockito.atLeast  
  
@RunWith(RobolectricTestRunner::class)  
class AddTipViewModelTest {  
  
 val mockPayment = Mockito.mock(Payment::class.*java*)  
 var viewModel: AddTipViewModel? = null  
 var payment = TestFactory.payment()  
  
 @Mock  
 lateinit var tipsUseCase: TipsUseCase  
 lateinit var tips: List<Tip>  
 lateinit var tipsPercent: List<Tip>  
  
 @Before  
 fun setUp() {  
 MockitoAnnotations.initMocks(this)  
 Mockito.`when`(mockPayment.total()).thenReturn(BigDecimal.valueOf(10L))  
 Mockito.`when`(mockPayment.subtotal).thenReturn(BigDecimal.valueOf(10L))  
 Mockito.`when`(mockPayment.tipableAmount()).thenReturn(BigDecimal.valueOf(10L))  
 Mockito.`when`(mockPayment.tip(Mockito.anyBoolean())).thenReturn(BigDecimal.valueOf(0L))  
  
 tips = *listOf*(  
 Tip(BigDecimal.*TEN*, false),  
 Tip(BigDecimal.*ONE*, false),  
 Tip(BigDecimal.*ZERO*, false)  
 )  
 Mockito.`when`(tipsUseCase.tipPresets()).thenReturn(tips)  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(false)  
 Mockito.`when`(tipsUseCase.isDisplayCalculatedAmountsEnabled()).thenReturn(false)  
  
 viewModel = AddTipViewModel(mockPayment,tipsUseCase)  
 }  
  
 @After  
 fun tearDown() {  
  
 }  
  
 @Test  
 fun testSetupTipWithoutCustomTip() {  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip1Text.*value*, CurrencyFormatter.format(tips[0].tip))  
 assertEquals(viewModel!!.tip2Text.*value*, CurrencyFormatter.format(tips[1].tip))  
  
 assertEquals(viewModel!!.tip1Visibility.*value*, View.*VISIBLE*)  
 assertEquals(viewModel!!.tip2Visibility.*value*, View.*VISIBLE*)  
 assertEquals(viewModel!!.tip3Visibility.*value*, View.*GONE*)  
 assertEquals(viewModel!!.customTipVisibility.*value*, View.*GONE*)  
 }  
  
 @Test  
 fun testSetupTipWithCustomTip() {  
  
 Mockito.`when`(tipsUseCase.isAllowCustomTipEnabled()).thenReturn(true)  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip1Text.*value*, CurrencyFormatter.format(tips[0].tip))  
  
 assertEquals(viewModel!!.tip1Visibility.*value*, View.*VISIBLE*)  
 assertEquals(viewModel!!.tip2Visibility.*value*, View.*VISIBLE*)  
 assertEquals(viewModel!!.tip3Visibility.*value*, View.*GONE*)  
 assertEquals(viewModel!!.customTipVisibility.*value*, View.*VISIBLE*)  
 }  
  
 @Test  
 fun testTip1ButtonTextAmount() {  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip1Text.*value*, CurrencyFormatter.format(tips[0].tip))  
  
 }  
  
 @Test  
 fun testTip2ButtonTextAmount() {  
 viewModel!!.setupTip(tipsUseCase)  
 assertEquals(CurrencyFormatter.format(tips[0].tip),viewModel!!.tip1Text.*value*)  
 }  
  
 @Test  
 fun testTip3ButtonTextAmount() {  
 tips = *listOf*(  
 Tip(BigDecimal.*TEN*, false),  
 Tip(BigDecimal.*ONE*, false),  
 Tip(BigDecimal.*TEN*, false)  
 )  
 Mockito.`when`(tipsUseCase.tipPresets()).thenReturn(tips)  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(false)  
 Mockito.`when`(tipsUseCase.isDisplayCalculatedAmountsEnabled()).thenReturn(false)  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip3Text.*value*, CurrencyFormatter.format(tips[2].tip))  
  
 }  
  
  
 @Test  
 fun testTip1ButtonTextPercent() {  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(true)  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip1Text.*value*, PercentFormatter.format(tips[0].tip))  
  
 }  
  
 @Test  
 fun testTip2ButtonTextPercent() {  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(true)  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip2Text.*value*, PercentFormatter.format(tips[1].tip))  
  
 }  
  
  
 @Test  
 fun testTip3ButtonTextPercent() {  
  
 tips = *listOf*(  
 Tip(BigDecimal.*TEN*, true),  
 Tip(BigDecimal.*ONE*, true),  
 Tip(BigDecimal("0.89"), true)  
 )  
 Mockito.`when`(tipsUseCase.tipPresets()).thenReturn(tips)  
  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(true)  
 viewModel!!.setupTip(tipsUseCase)  
  
 assertEquals(viewModel!!.tip3Text.*value*, PercentFormatter.format(tips[2].tip))  
  
 }  
  
  
 @Test  
 fun testTip1ButtonTextPercentWithCalculatedAmount() {  
  
 Mockito.`when`(mockPayment.tip(false)).thenReturn(BigDecimal("1.00"))  
  
 tipsPercent = *listOf*(  
 Tip(BigDecimal("0.1"), true),  
 Tip(BigDecimal("0.01"), true),  
 Tip(BigDecimal("0.01"), true)  
 )  
  
 Mockito.`when`(tipsUseCase.tipPresets()).thenReturn(tipsPercent)  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(true)  
 Mockito.`when`(tipsUseCase.isDisplayCalculatedAmountsEnabled()).thenReturn(true)  
  
 viewModel!!.setupTip(tipsUseCase)  
 assertEquals(expectedCalculatedButtonText(tipsPercent[0].tip),viewModel!!.tip1Text.*value*)  
 }  
  
 @Test  
 fun testTip2ButtonTextPercentWithCalculatedAmount() {  
  
 Mockito.`when`(mockPayment.tip(false)).thenReturn(BigDecimal("9.90"))  
  
 tipsPercent = *listOf*(  
 Tip(BigDecimal("0.1"), true),  
 Tip(BigDecimal("0.99"), true),  
 Tip(BigDecimal("0.01"), true)  
 )  
  
 Mockito.`when`(tipsUseCase.tipPresets()).thenReturn(tipsPercent)  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(true)  
 Mockito.`when`(tipsUseCase.isDisplayCalculatedAmountsEnabled()).thenReturn(true)  
  
 viewModel!!.setupTip(tipsUseCase)  
 assertEquals(expectedCalculatedButtonText(tipsPercent[1].tip),viewModel!!.tip2Text.*value*)  
  
 }  
  
 @Test  
 fun testTip3ButtonTextPercentWithCalculatedAmount() {  
  
 Mockito.`when`(mockPayment.tip(false)).thenReturn(BigDecimal("0.80"))  
  
 tipsPercent = *listOf*(  
 Tip(BigDecimal("0.1"), true),  
 Tip(BigDecimal("0.99"), true),  
 Tip(BigDecimal("0.08"), true)  
 )  
  
 Mockito.`when`(tipsUseCase.tipPresets()).thenReturn(tipsPercent)  
 Mockito.`when`(tipsUseCase.isTipInPercent()).thenReturn(true)  
 Mockito.`when`(tipsUseCase.isDisplayCalculatedAmountsEnabled()).thenReturn(true)  
  
 viewModel!!.setupTip(tipsUseCase)  
 assertEquals(expectedCalculatedButtonText(tipsPercent[2].tip),viewModel!!.tip3Text.*value*)  
  
 }  
  
 @Test  
 fun testNoTipButtonSetTipZero() {  
 viewModel!!.setTipPercent(R.id.*noTipButton*)  
 Mockito.verify(mockPayment, atLeast(2)).setTip(BigDecimal.*ZERO*, false)  
 }  
  
 @Test  
 fun testSetTipPercentWithValidId() {  
 viewModel!!.setTipPercent(R.id.*tipOneButton*)  
 Mockito.verify(mockPayment, atLeast(2)).setTip(tips[0].tip, false)  
 }  
  
 @Test  
 fun testSetTipPercentWithInvalidId() {  
 viewModel!!.setTipPercent(-2)  
 Mockito.verify(mockPayment).setTip(BigDecimal.*ZERO*, false)  
 }  
  
 @Test  
 fun testSetTip() {  
 viewModel!!.setTip(BigDecimal.*TEN*, true)  
 Mockito.verify(mockPayment).setTip(BigDecimal.*TEN*, true)  
 }  
  
 @Test  
 fun testUpdateAmountsNonZeroTip() {  
 Mockito.`when`(mockPayment.tip(Mockito.anyBoolean())).thenReturn(BigDecimal.*TEN*)  
  
 viewModel!!.updateAmounts()  
  
 val correctFormat = CurrencyFormatter.format(mockPayment.tipableAmount()) +  
 " + " + CurrencyFormatter.format(mockPayment.tip(false)) +  
 " tip"  
  
 assertEquals(viewModel!!.tipBreakdownVisibility.*value*, View.*VISIBLE*)  
 assertEquals(viewModel!!.tipBreakdown.*value*, correctFormat)  
 assertEquals(viewModel!!.amountLabel.*value*, CurrencyFormatter.format(mockPayment.total()))  
 }  
  
 @Test  
 fun testUpdateAmountsZeroTip() {  
 Mockito.`when`(mockPayment.tip(Mockito.anyBoolean())).thenReturn(BigDecimal.*ZERO*)  
  
 viewModel!!.updateAmounts()  
  
 assertEquals(viewModel!!.tipBreakdownVisibility.*value*, View.*GONE*)  
 assertEquals(viewModel!!.amountLabel.*value*, CurrencyFormatter.format(mockPayment.total()))  
 }  
  
 private fun expectedCalculatedButtonText(tip : BigDecimal): String{  
 payment.setTip(tip,true)  
 return PercentFormatter.format(tip) + "\n" + CurrencyFormatter.format(payment.tip(false))  
  
 }  
}