**Gradle.build file**

***/\****

***\* This file was generated by the Gradle 'init' task.***

***\****

***\* This generated file contains a sample Java Library project to get you started.***

***\* For more details take a look at the Java Libraries chapter in the Gradle***

***\* User Manual available at https://docs.gradle.org/6.3/userguide/java\_library\_plugin.html***

***\*/***

***plugins {***

***id 'checkstyle'***

***// Apply the java-library plugin to add support for Java Library***

***id 'java-library-distribution'***

***}***

***group ‘tradeFinance’***

***version '1.0'***

***repositories {***

***// Use jcenter for resolving dependencies.***

***// You can declare any Maven/Ivy/file repository here.***

***maven {***

***url "https://hyperledger.jfrog.io/hyperledger/fabric-maven"***

***}***

***jcenter()***

***maven {***

***url 'https://jitpack.io'***

***}***

***}***

***dependencies {***

***compileOnly 'org.hyperledger.fabric-chaincode-java:fabric-chaincode-shim:2.2.+'***

***implementation 'com.owlike:genson:1.5'***

***// This dependency is exported to consumers, that is to say found on their compile classpath.***

***api 'org.apache.commons:commons-math3:3.6.1'***

***// This dependency is used internally, and not exposed to consumers on their own compile classpath.***

***implementation 'com.google.guava:guava:28.2-jre'***

***// Use JUnit test framework***

***testImplementation 'junit:junit:4.12'***

***}***

**Letter of credit for variables**

**Name of the class is : lcVariables.java**

***package tradeFinance;***

***import com.owlike.genson.annotation.JsonProperty;***

***import org.hyperledger.fabric.contract.annotation.DataType;***

***import org.hyperledger.fabric.contract.annotation.Property;***

***import java.util.Objects;***

***@DataType()***

***public final class lcVariable {***

***@Property()***

***private final String id;***

***@Property()***

***private final String expiryDate;***

***@Property()***

***private final String buyer;***

***@Property()***

***private final String bank;***

***@Property()***

***private final String seller;***

***@Property()***

***private final String amount;***

***@Property()***

***private final String status;***

***public lcVariable(@JsonProperty("id") final String id,***

***@JsonProperty("expiryDate") final String expiryDate,***

***@JsonProperty("buyer") final String buyer,***

***@JsonProperty("bank") final String bank,***

***@JsonProperty("seller") final String seller,***

***@JsonProperty("amount") final String amount,***

***@JsonProperty("status") final String status) {***

***this.id = id;***

***this.expiryDate = expiryDate;***

***this.buyer = buyer;***

***this.bank = bank;***

***this.seller = seller;***

***this.amount = amount;***

***this.status = status;***

***}***

***public String getId() {***

***return id;***

***}***

***public String getExpiryDate() {***

***return expiryDate;***

***}***

***public String getBuyer() {***

***return buyer;***

***}***

***public String getBank() {***

***return bank;***

***}***

***public String getSeller() {***

***return seller;***

***}***

***public String getAmount() {***

***return amount;***

***}***

***public String getStatus() {***

***return status;***

***}***

***@Override***

***public boolean equals(final Object obj) {***

***if (this == obj) {***

***return true;***

***}***

***if ((obj == null) || (getClass() != obj.getClass())) {***

***return false;***

***}***

***lcVariable other = (lcVariable) obj;***

***return Objects.deepEquals(new String[] { getId(), getExpiryDate(), getBuyer(), getBank(), getSeller(), getAmount(), getStatus() },***

***new String[] { other.getId(), other.getExpiryDate(), other.getBuyer(), other.getBank(), other.getSeller(), other.getAmount(), other.getStatus() });***

***}***

***@Override***

***public int hashCode() {***

***return Objects.hash(getId(), getExpiryDate(), getBuyer(), getBank(), getSeller(), getAmount(), getStatus());***

***}***

***@Override***

***public String toString() {***

***return this.getClass().getSimpleName() + "@" + Integer.toHexString(hashCode()) + " [id=" + id + ", expiryDate=" + expiryDate + ", buyer=" + buyer + ", bank=" + bank + ", seller=" + seller + ", amount=" + amount + ", status=" + status + "]";***

***}***

***}***

**Trade Finance Functions**

**Name of class is : tradeFinanceFunction.java**

***package tradeFinance;***

***import org.hyperledger.fabric.contract.Context;***

***import org.hyperledger.fabric.contract.ContractInterface;***

***import org.hyperledger.fabric.contract.annotation.Contract;***

***import org.hyperledger.fabric.contract.annotation.Default;***

***import org.hyperledger.fabric.contract.annotation.Info;***

***import org.hyperledger.fabric.contract.annotation.Transaction;***

***import org.hyperledger.fabric.shim.ChaincodeException;***

***import org.hyperledger.fabric.shim.ChaincodeStub;***

***import com.owlike.genson.Genson;***

***@Contract(***

***name = "tradeFinance",***

***info = @Info(***

***title = "Trade Finance Contract",***

***description = "A Sample letter of credit/Trade finance chaincode example",***

***version = "0.0.1-SNAPSHOT"))***

***@Default***

***public final class TradeFinanceFunction implements ContractInterface {***

***private final Genson genson = new Genson();***

***private enum TradeFinanceErrors {***

***LC\_NOT\_FOUND,***

***LC\_ALREADY\_EXISTS***

***}***

***@Transaction()***

***public void initLedger(final Context ctx) {***

***ChaincodeStub stub = ctx.getStub();***

***lcVariable lcVariable = new lcVariable("1", "30/march/2024", "Zen", "Citi Bank", "Raja", "100000", "Lc Not Created");***

***String lcVariableState = genson.serialize(lcVariable);***

***stub.putStringState("1", lcVariableState);***

***}***

***@Transaction()***

***public lcVariable requestLetterOfCredit(final Context ctx, final String id, final String expiryDate, final String buyer, final String bank, final String seller, final String amount, final String status) {***

***ChaincodeStub stub = ctx.getStub();***

***String lcVariableState = stub.getStringState(id);***

***if (!lcVariableState.isEmpty()) {***

***String errorMessage = String.format("LC already exists with ID: %s", id);***

***System.out.println(errorMessage);***

***throw new ChaincodeException(errorMessage, TradeFinanceErrors.LC\_ALREADY\_EXISTS.toString());***

***}***

***lcVariable lcVariable = new lcVariable(id, expiryDate, buyer, bank, seller, amount, status);***

***lcVariableState = genson.serialize(lcVariable);***

***stub.putStringState(id, lcVariableState);***

***return lcVariable;***

***}***

***@Transaction()***

***public lcVariable viewLc(final Context ctx, final String id) {***

***ChaincodeStub stub = ctx.getStub();***

***String lcVariableState = stub.getStringState(id);***

***if (lcVariableState.isEmpty()) {***

***String errorMessage = String.format("LC does not exist with ID: %s", id);***

***System.out.println(errorMessage);***

***throw new ChaincodeException(errorMessage, TradeFinanceErrors.LC\_NOT\_FOUND.toString());***

***}***

***lcVariable lcVariable = genson.deserialize(lcVariableState, lcVariable.class);***

***return lcVariable;***

***}***

***@Transaction()***

***public lcVariable issueLc(final Context ctx, final String id, final String newStatus) {***

***ChaincodeStub stub = ctx.getStub();***

***String lcVariableState = stub.getStringState(id);***

***if (lcVariableState.isEmpty()) {***

***String errorMessage = String.format("LC does not exist with ID: %s", id);***

***System.out.println(errorMessage);***

***throw new ChaincodeException(errorMessage, TradeFinanceErrors.LC\_NOT\_FOUND.toString());***

***}***

***lcVariable lcVariable = genson.deserialize(lcVariableState, lcVariable.class);***

***lcVariable newLcVariable = new lcVariable(lcVariable.getId(), lcVariable.getExpiryDate(), lcVariable.getBuyer(), lcVariable.getBank(), lcVariable.getSeller(), lcVariable.getAmount(), newStatus);***

***String newLcVariableState = genson.serialize(newLcVariable);***

***stub.putStringState(id, newLcVariableState);***

***return newLcVariable;***

***}***

***@Transaction()***

***public lcVariable acceptLc(final Context ctx, final String id, final String newStatus) {***

***ChaincodeStub stub = ctx.getStub();***

***String lcVariableState = stub.getStringState(id);***

***if (lcVariableState.isEmpty()) {***

***String errorMessage = String.format("LC does not exist with ID: %s", id);***

***System.out.println(errorMessage);***

***throw new ChaincodeException(errorMessage, TradeFinanceErrors.LC\_NOT\_FOUND.toString());***

***}***

***lcVariable lcVariable = genson.deserialize(lcVariableState, lcVariable.class);***

***lcVariable newLcVariable = new lcVariable(lcVariable.getId(), lcVariable.getExpiryDate(), lcVariable.getBuyer(), lcVariable.getBank(), lcVariable.getSeller(), lcVariable.getAmount(), newStatus);***

***String newLcVariableState = genson.serialize(newLcVariable);***

***stub.putStringState(id, newLcVariableState);***

***return newLcVariable;***

***}***

***}***

***Command lines used to invoke functions :***

***1.Request Letter Of Credit***

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n tradeFinance --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["requestLetterOfCredit", "2", "25/march/2024","WallMart","J P Morgan","Dmart","10000","Pending"]}'

***2.viewLc:***

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n tradeFinance --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["viewLc","2"]}'

***3.issueLc:***

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n tradeFinance --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["issueLc","2","Approved"]}'

***4.AcceptLc:***

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C mychannel -n tradeFinance --peerAddresses localhost:7051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG1 --peerAddresses localhost:9051 --tlsRootCertFiles $CORE\_PEER\_TLS\_ROOTCERT\_FILE\_ORG2 -c '{"Args":["acceptLc","2","Accepted"]}'