package player.playeragent;

import player.*;
import util.BreakEven;
import edu.neu.ccs.demeterf.demgen.lib.List;
import gen.);
import java.util.Random;

/** Class for creating a derivative */
public class CreateAgent implements PlayerI.CreateAgentI{

  /** Returns a newly created derivative of a different type than already existing derivatives */
  public Derivative createDerivative(Player player, List<Type> existing){
    Random rand = new Random();
    Type type;
    //Keep making Types unless the type we made is unique
    /*do{
      type = new Type(getRelations());
    }while(existing.contains(type));
    */
    //Make fake derivative
    type = new Type(getRelations());
    Derivative d = new Derivative(Util.freshName(player), player.id, new Price(0), type);
    double breakEven = 1;
    //Calculate the Break-Even Point for our Derivative
    try{
      breakEven = BreakEven.getBreakEvenPoint(BreakEven.getRelations(d));
    }catch(Exception e){}
    //Find the complement of the break-even price * a double + the break even price
    double priceNum = breakEven + ((1.0 - breakEven)*rand.nextDouble());
    Price price = new Price(priceNum);
    //Create and Return the Derivative
    return new Derivative(Util.freshName(player), player.id, price, type);
  }

  /**
   * @return List<TypeInstance> - Returns a list of TypeInstances of length 1 - 10
   */
  private static List<TypeInstance> getRelations(){
    Random rand = new Random();
    int relCount = rand.nextInt(256)+1;
    List<TypeInstance> typeList = List.create();
    for (int index=0; index < relCount; index++){
      typeList = typeList.push(getRelationNumber());
    }
    /*For Debugging
    java.util.Iterator<TypeInstance> iter = typeList.iterator();
    String types = "";
    while(iter.hasNext()){/*
      types += (" .next().print () + " ");
    }
    System.out.println("Finished getRelations(): " + types + "\" + System.currentTimeMillis());
    */
    return typeList;
  }

  /**
   * Generates an Instance Type with a Relation Number in it generated
   * by a random number generator.
   * @return TypeInstance - A type instance with an even relation number between 0-126
   */
  private static TypeInstance getRelationNumber(){
    Random rand = new Random();
    return new TypeInstance(new RelationNr(rand.nextInt(64)*2));
  }
}