```java
package player;

import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

import edu.neu.ccs.demeterf.demfgen.lib.List;
import edu.neu.ccs.satsolver.InputInitialI;
import gen.*;

public class InputInitial implements InputInitialI {
	public Derivative d;
	public Set<PairIt> p;

	public InputInitial(Derivative d){
	this.d = d;
	this.p = this.getPairs();
	}

	@Override
	public Set<PairIt> getPairs() {
		Set<PairIt> s = new HashSet<PairIt>();
		Set<RelationNr> rs = new HashSet<RelationNr>();
		double fraction;
		double[] bigArray = new double[256];
		List<Constraint> loc = d.optraw.inner().instance.cs;
		Iterator<Constraint> iter = loc.iterator();
		Constraint c;

		double totalweight = 0.0;

		while(iter.hasNext()){
			c = iter.next();
			totalweight += c.w.v;
			bigArray[c.r.v] += (double)c.w.v;
			rs.add(c.r);
		}

		/*
		for(int i=0; i<loc.length(); i++){
			c = loc.lookup(i);
			totalweight += c.w.v;
			bigArray[c.r.v] += (double)c.w.v;	//keep track of weights
			rs.add(c.r);
				//make a set of unique relation numbers
		}
		*/
		Iterator<RelationNr> r_iter = rs.iterator();
		while(r_iter.hasNext()){
			RelationNr r = (RelationNr) r_iter.next();
			fraction = bigArray[r.v]/totalweight;
			//System.out.println(fraction);
			s.add(new PairIt(r, fraction));
		}

		return s;
	}

}
```