<<abstract>> Competition - match: Match - final competitors: List<Competitor> Diageamme UML projet COO -observers : List<Observer> Competitor -RACHEDI ILYAS <<constructor>> - name: String -SAADI NADINE + Competition(List<Competitor>,Match, - points: int List<Observer) <<constructor>> <<methods>> +Competitor(String) + play() +getPoints():int # play(List<Competitor>) +getName():String # playMatch(Competitor, Competitor) +addPoint() +ranking():Map<Competitor,int> +createCopy(List<Competitor>): List<Competitor> +winner(): Competitor + setToZero(List<Competitor>) : void + getObservers(): List<Observer> + getCompetitors () : List<Competitor> + getRank (): Map<Competitor ,Integer> + addObservers() : void + getMatch(): Match **Tournament** Master League -strat: Strategy -groups: int <<constructor>> <<constructor>> + Tournament(List<Competitor>,Match) +League(List<Competitor>,Match) <<methods>> <<constructor>> <<methods>> #play(List<Competitor>) +Master(List<Competitor>,Match,Strategy,groups) #play(List<Competitor>) -removeLosers(List<Competitor>,int):List<Competitor> <<methods>> -isPowerOf2():boolean #play(List<Competitor>) + poolPhase(List<Competitor>):List<Competitor> +cutToGroups(List<Competitor>): List<List<Competitor>> + getGroups(): int <<interface>> <<abstract>> Match Strategy -Competitor winner -Competitor looser + chooseWinner(List<Map<Competitor, Integer>>): List<Competitor> -Competitor first -Competitor second -List<Observer> observers <<constructor>> -implements (interface)implements(interface) +Match() -implements(interface)-<<methods>> #winMatch(Competitor,Competitor):Competitor RandomStrategy + getWinner() : Competitor HumanInputStrategy + getLooser() : Competitor -size:int -size:int + getObservers : List<Observer> **BasicStrategy** -n: int - n : int + getFirst() : Competitor -generator : static Random + getSecond(): Compteitor default = new Random(int) + setWinner(Comptitor) : void <<constructor>> + setLooser(Comptitor) : void + HumanInputStrategy(int) <<constructor>> + addObserver(Observer) : void <<constructor>> <<methods>> + BasicStrategy(int) + RandomStrategy(int) + removeObserver(Observer) : void <<methods>> <<methods>> + chooseWinner(List<Map<Competitor, Integer>>): List<Competitor + fireEndMatch(Match): void r(List<Map<Competitor, Integer>>): List<Competitor Texte MatchRandom <<interface>> Observer <<constructor>> +MatchRandom() + reactToMatch():void <<methods>> #winMatch(Competitor,Competitor):Competitor -implement--implement-BookMaker Journalist - comp:Competition - name:String - name:String <<constructor>> <<constructor>> + Journalist (String) +BookMaker(String, Competition)

<<methodes>>

+ reactToMatch():void

<methodes>>

+ reactToMatch():void