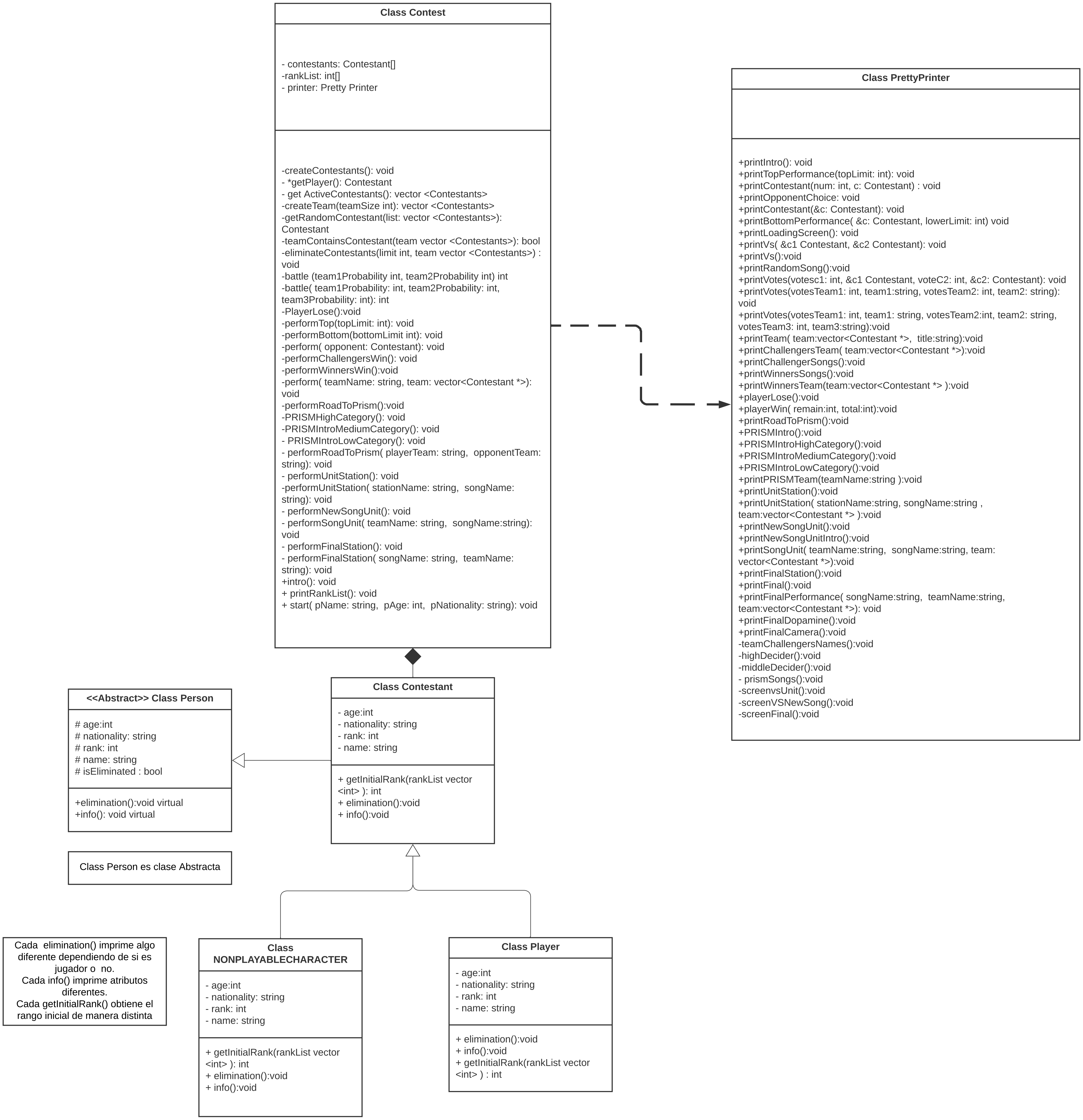

Class Contest

- contestants: Contestant[]
-rankList: int[]
- printer: Pretty Printer

-createContestants(): void
- *getPlayer(): Contestant
- get ActiveContestants(): vector <Contestants>
-createTeam(teamSize int): vector <Contestants>
-getRandomContestant(list: vector <Contestants>): Contestant
-teamContainsContestant(team vector <Contestants>): bool
-eliminateContestants(limit int, team vector <Contestants>) : void
-battle (team1Probability int, team2Probability int) int
-battle(team1Probability: int, team2Probability: int, team3Probability: int): int
-PlayerLose():void
-performTop(topLimit: int): void
-performBottom(bottomLimit int): void
-perform(opponent: Contestant): void
-performChallengersWin(): void
-performWinnersWin():void
-perform(teamName: string, team: vector<Contestant *>): void
-performRoadToPrism():void
-PRISMHighCategory(): void
-PRISMIntroMediumCategory(): void
- PRISMIntroLowCategory(): void
- performRoadToPrism(playerTeam: string, opponentTeam: string): void
- performUnitStation(): void
-performUnitStation(stationName: string, songName: string): void
- performNewSongUnit(): void
- performSongUnit(teamName: string, songName:string): void
- performFinalStation(): void
- performFinalStation(songName: string, teamName: string): void
+intro(): void
+ printRankList(): void
+ start(pName: string, pAge: int, pNationality: string): void

Class PrettyPrinter

+printIntro(): void
+printTopPerformance(topLimit: int): void
+printContestant(num: int, c: Contestant) : void
+printOpponentChoice: void
+printContestant(&c: Contestant): void
+printBottomPerformance(&c: Contestant, lowerLimit: int) void
+printLoadingScreen(): void
+printVs(&c1 Contestant, &c2 Contestant): void
+printVs():void
+printRandomSong():void
+printVotes(votesc1: int, &c1 Contestant, voteC2: int, &c2: Contestant): void
+printVotes(votesTeam1: int, team1:string, votesTeam2: int, team2: string): void
+printVotes(votesTeam1: int, team1: string, votesTeam2:int, team2: string, votesTeam3: int, team3:string):void
+printTeam(team:vector<Contestant *>, title:string):void
+printChallengersTeam(team:vector<Contestant *>):void
+printChallengerSongs():void
+printWinnersSongs():void
+printWinnersTeam(team:vector<Contestant *>):void
+playerLose():void
+playerWin(remain:int, total:int):void
+printRoadToPrism():void
+PRISMIntro():void
+PRISMIntroHighCategory():void
+PRISMIntroMediumCategory():void
+PRISMIntroLowCategory():void
+printPRISMTeam(teamName:string):void
+printUnitStation():void
+printUnitStation(stationName:string, songName:string , team:vector<Contestant *>):void
+printNewSongUnit():void
+printNewSongUnitIntro():void
+printSongUnit(teamName:string, songName:string, team: vector<Contestant *>):void
+printFinalStation():void
+printFinal():void
+printFinalPerformance(songName:string, teamName:string, team:vector<Contestant *>): void
+printFinalDopamine():void
+printFinalCamera():void
-teamChallengersNames():void
-highDecider():void
-middleDecider():void
- prismSongs():void
-screenvsUnit():void
-screenVSNewSong():void
-screenFinal():void

<<Abstract>> Class Person

# age:int
# nationality: string
# rank: int
# name: string
# isEliminated : bool

+elimination():void virtual
+info(): void virtual

Class Contestant

- age:int
- nationality: string
- rank: int
- name: string

+ getInitialRank(rankList vector <int>): int
+ elimination():void
+ info():void

Class Person es clase Abstracta

Cada elimination() imprime algo diferente dependiendo de si es jugador o no.
Cada info() imprime atributos diferentes.
Cada getInitialRank() obtiene el rango inicial de manera distinta

Class NONPLAYABLECHARACTER

- age:int
- nationality: string
- rank: int
- name: string

+ getInitialRank(rankList vector <int>): int
+ elimination():void
+ info():void

Class Player

- age:int
- nationality: string
- rank: int
- name: string

+ elimination():void
+ info():void
+ getInitialRank(rankList vector <int>) : int