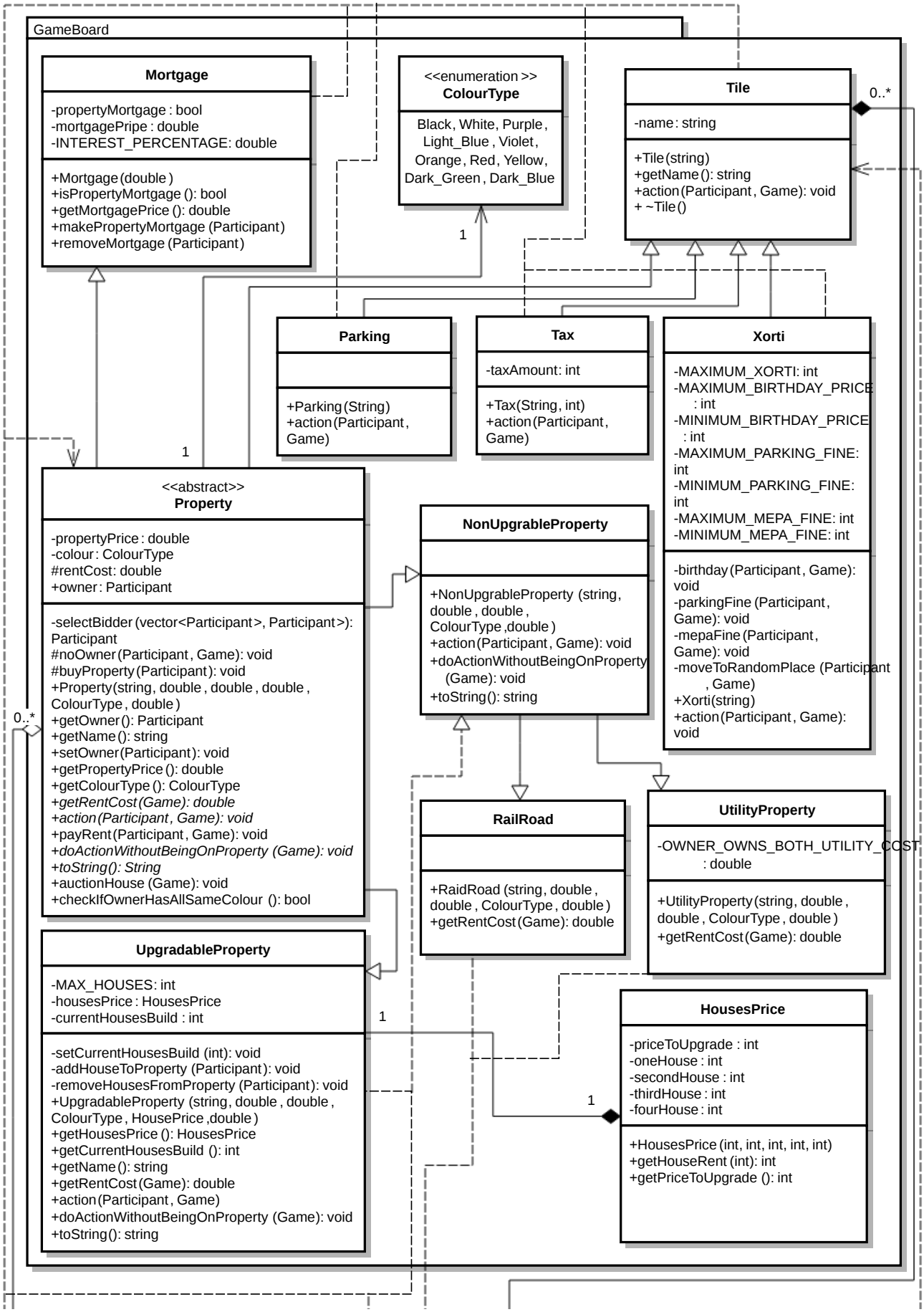
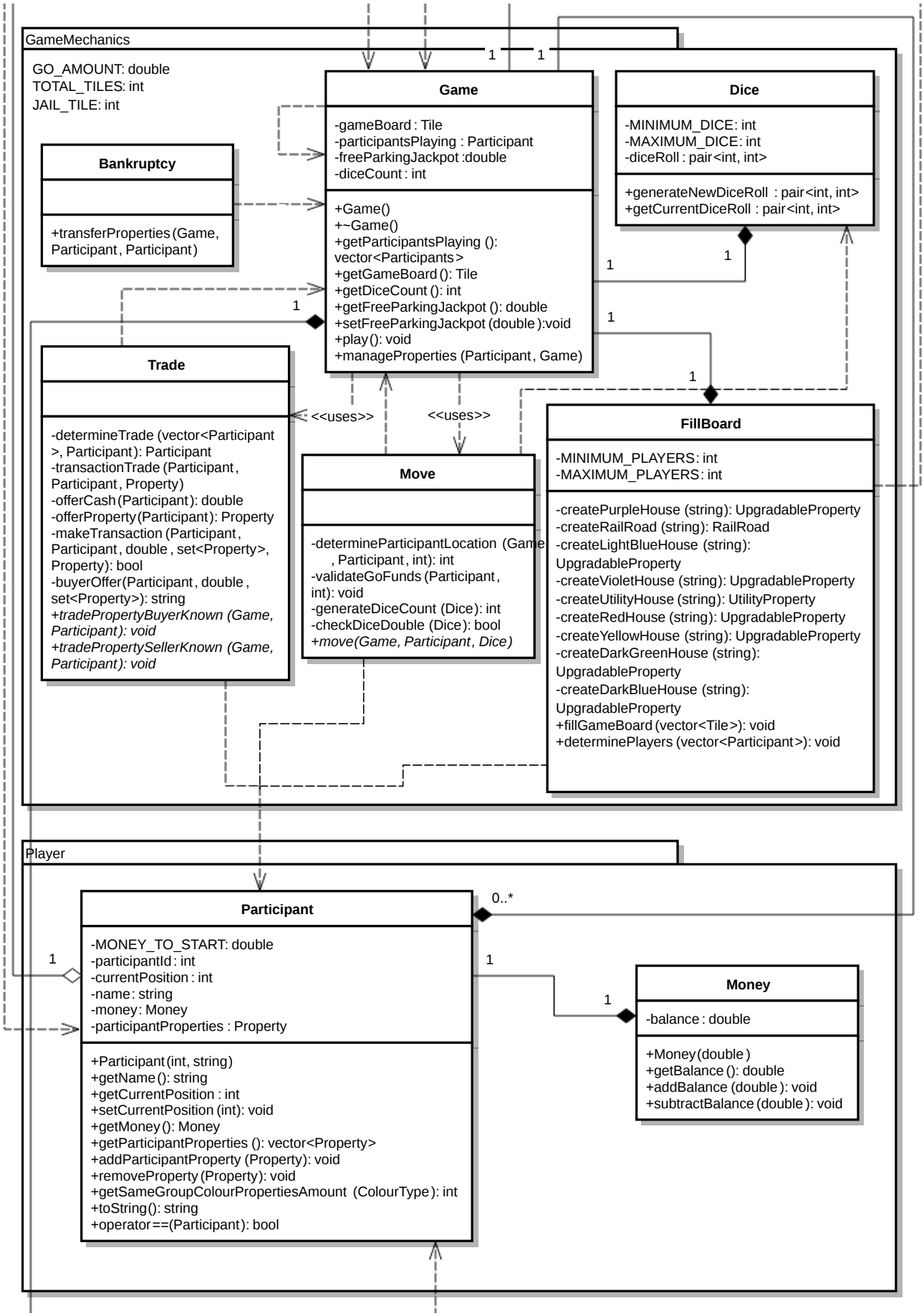
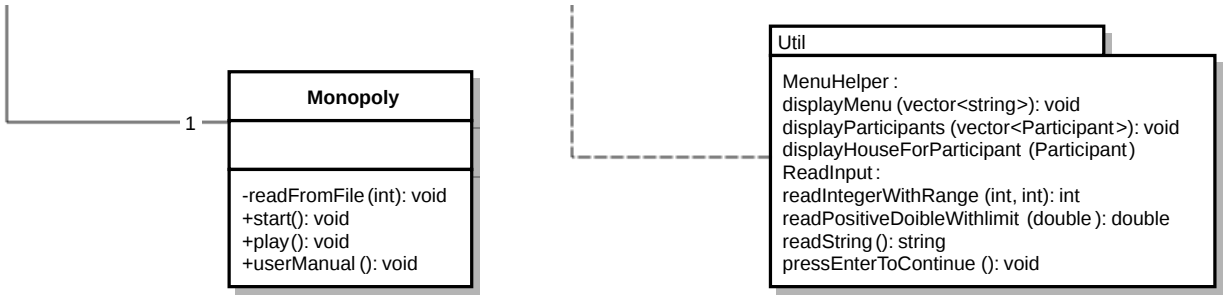

GameBoard
Mortgage
-propertyMortgage : bool
-mortgagePripe : double
-INTEREST_PERCENTAGE: double
+Mortgage (double)
+isPropertyMortgage (): bool
+getMortgagePrice (): double
+makePropertyMortgage (Participant)
+removeMortgage (Participant)
<<enumeration >>
ColourType
Black, White, Purple, Light_Blue , Violet, Orange, Red, Yellow, Dark_Green , Dark_Blue
Tile
-name: string
+Tile (string)
+getName (): string
+action (Participant , Game): void
+ ~Tile ()
0..*
1
Parking
+Parking (String)
+action (Participant , Game)
Tax
-taxAmount: int
+Tax(String, int)
+action (Participant , Game)
Xorti
-MAXIMUM_XORTI: int
-MAXIMUM_BIRTHDAY_PRICE : int
-MINIMUM_BIRTHDAY_PRICE : int
-MAXIMUM_PARKING_FINE: int
-MINIMUM_PARKING_FINE: int
-MAXIMUM_MEPA_FINE: int
-MINIMUM_MEPA_FINE: int
-birthday (Participant , Game): void
-parkingFine (Participant , Game): void
-mepaFine (Participant , Game): void
-moveToRandomPlace (Participant , Game)
+Xorti(string)
+action (Participant , Game): void
1
<<abstract>>
Property
-propertyPrice : double
-colour : ColourType
#rentCost: double
+owner : Participant
-selectBidder (vector<Participant>, Participant>): Participant
#noOwner (Participant , Game): void
#buyProperty (Participant): void
+Property(string, double , double , double , ColourType , double)
+getOwner (): Participant
+getName (): string
+setOwner (Participant): void
+getPropertyPrice (): double
+getColourType (): ColourType
+getRentCost (Game): double
+action (Participant , Game): void
+payRent (Participant , Game): void
+doActionWithoutBeingOnProperty (Game): void
+toString(): String
+auctionHouse (Game): void
+checkIfOwnerHasAllSameColour (): bool
0..*
NonUpgrableProperty
+NonUpgrableProperty (string, double , double , ColourType ,double)
+action (Participant , Game): void
+doActionWithoutBeingOnProperty (Game): void
+toString (): string
RailRoad
+RaidRoad (string, double , double , ColourType , double)
+getRentCost (Game): double
UtilityProperty
-OWNER_OWNS_BOTH_UTILITY_COST : double
+UtilityProperty (string, double , double , ColourType , double)
+getRentCost (Game): double
UpgradableProperty
-MAX_HOUSES: int
-housesPrice : HousesPrice
-currentHousesBuild : int
-setCurrentHousesBuild (int): void
-addHouseToProperty (Participant): void
-removeHousesFromProperty (Participant): void
+UpgradableProperty (string, double , double , ColourType , HousePrice ,double)
+getHousesPrice (): HousesPrice
+getCurrentHousesBuild (): int
+getName (): string
+getRentCost (Game): double
+action (Participant , Game)
+doActionWithoutBeingOnProperty (Game): void
+toString (): string
1
HousesPrice
-priceToUpgrade : int
-oneHouse : int
-secondHouse : int
-thirdHouse : int
-fourHouse : int
+HousesPrice (int, int, int, int, int)
+getHouseRent (int): int
+getPriceToUpgrade (): int
1


GameMechanics
GO_AMOUNT: double
TOTAL_TILES: int
JAIL_TILE: int
Bankruptcy
+transferProperties(Game, Participant, Participant)
Game
-gameBoard : Tile
-participantsPlaying : Participant
-freeParkingJackpot :double
-diceCount : int
+Game()
+~Game()
+getParticipantsPlaying (): vector<Participants>
+getGameBoard (): Tile
+getDiceCount (): int
+getFreeParkingJackpot (): double
+setFreeParkingJackpot (double):void
+play (): void
+manageProperties (Participant, Game)
Dice
-MINIMUM_DICE: int
-MAXIMUM_DICE: int
-diceRoll : pair<int, int>
+generateNewDiceRoll : pair<int, int>
+getCurrentDiceRoll : pair<int, int>
Trade
-determineTrade (vector<Participant>, Participant): Participant
-transactionTrade (Participant, Participant, Property)
-offerCash (Participant): double
-offerProperty (Participant): Property
-makeTransaction (Participant, Participant, double, set<Property>, Property): bool
-buyerOffer(Participant, double, set<Property>): string
+tradePropertyBuyerKnown (Game, Participant): void
+tradePropertySellerKnown (Game, Participant): void
<<uses>>
<<uses>>
Move
-determineParticipantLocation (Game, Participant, int): int
-validateGoFunds (Participant, int): void
-generateDiceCount (Dice): int
-checkDiceDouble (Dice): bool
+move(Game, Participant, Dice)
FillBoard
-MINIMUM_PLAYERS: int
-MAXIMUM_PLAYERS: int
-createPurpleHouse (string): UpgradableProperty
-createRailRoad (string): RailRoad
-createLightBlueHouse (string): UpgradableProperty
-createVioletHouse (string): UpgradableProperty
-createUtilityHouse (string): UtilityProperty
-createRedHouse (string): UpgradableProperty
-createYellowHouse (string): UpgradableProperty
-createDarkGreenHouse (string): UpgradableProperty
-createDarkBlueHouse (string): UpgradableProperty
+fillGameBoard (vector<Tile>): void
+determinePlayers (vector<Participant>): void
Player
Participant
-MONEY_TO_START: double
-participantId : int
-currentPosition : int
-name : string
-money : Money
-participantProperties : Property
+Participant (int, string)
+getName (): string
+getCurrentPosition : int
+setCurrentPosition (int): void
+getMoney (): Money
+getParticipantProperties (): vector<Property>
+addParticipantProperty (Property): void
+removeProperty (Property): void
+getSameGroupColourPropertiesAmount (ColourType): int
+toString (): string
+operator==(Participant): bool
Money
-balance : double
+Money (double)
+getBalance (): double
+addBalance (double): void
+subtractBalance (double): void
0..*
1
1
1
1
1
1
1
1
1


1
Monopoly
-readFromFile (int): void
+start(): void
+play (): void
+userManual (): void
Util
MenuHelper :
displayMenu (vector<string>): void
displayParticipants (vector<Participant >): void
displayHouseForParticipant (Participant)
ReadInput :
readIntegerWithRange (int, int): int
readPositiveDoibleWithlimit (double): double
readString (): string
pressEnterToContinue (): void