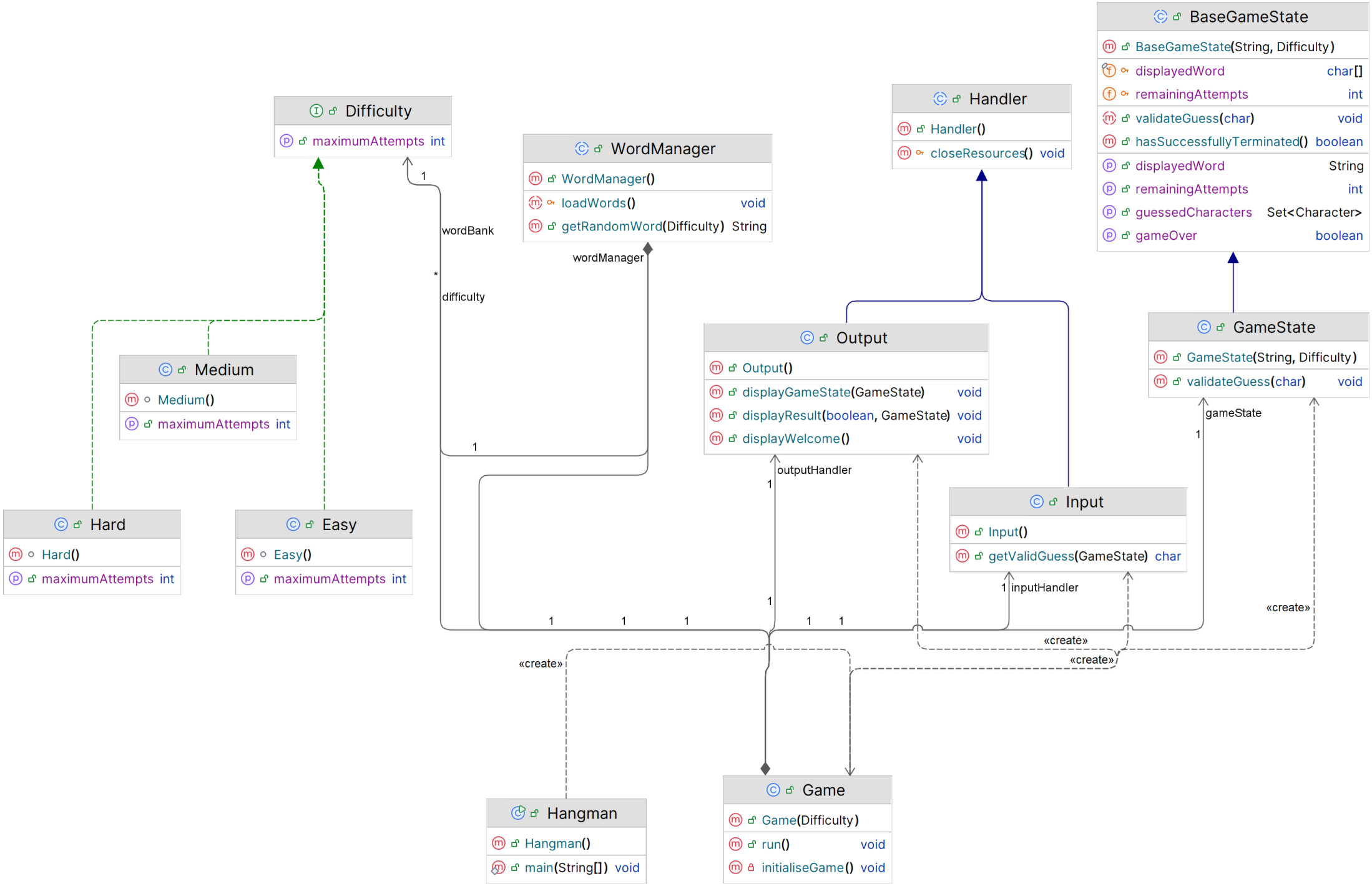
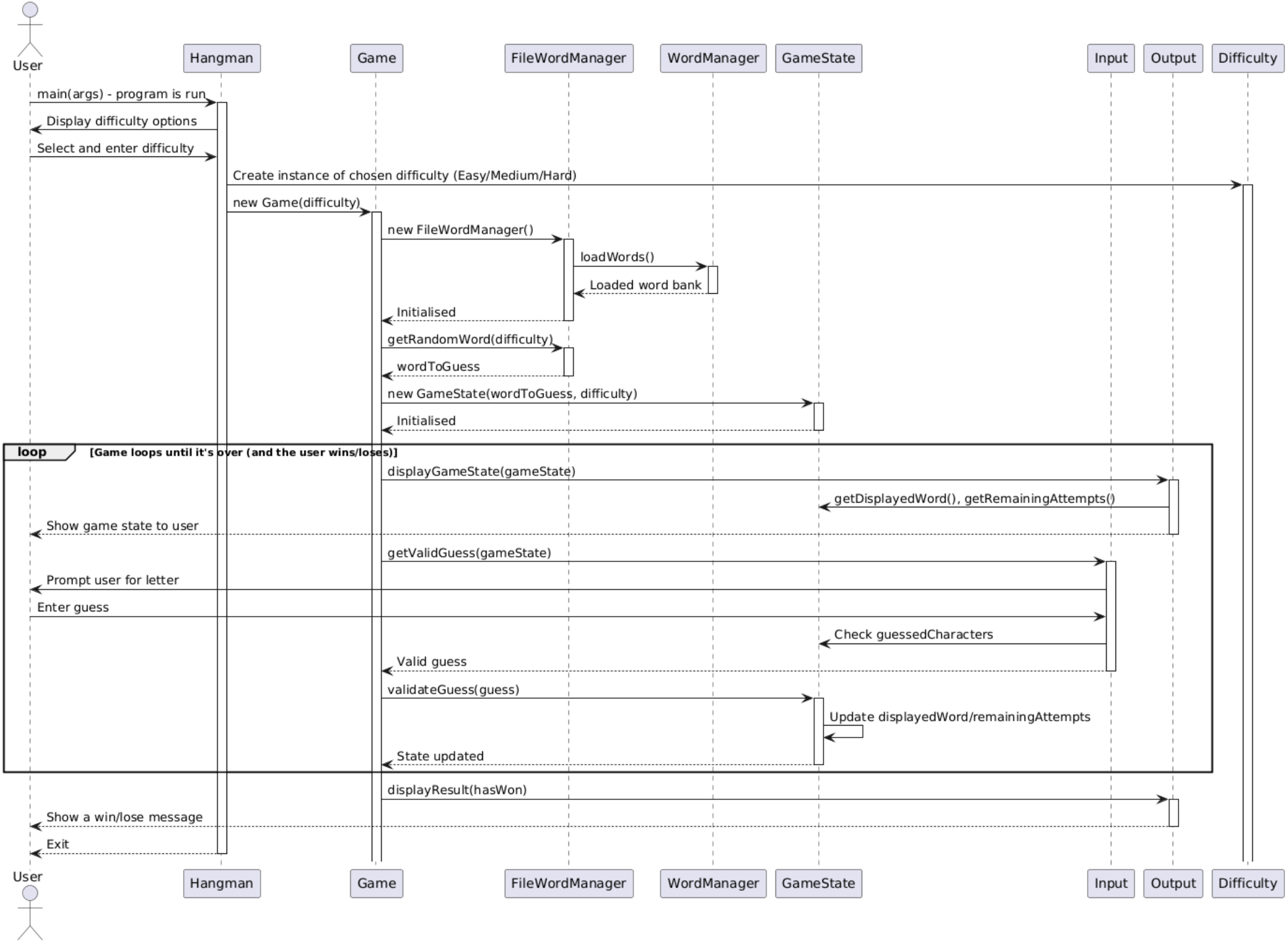

Difficulty
maximumAttempts int
WordManager
WordManager()
loadWords() void
getRandomWord(Difficulty) String
Handler
Handler()
closeResources() void
BaseGameState
BaseGameState(String, Difficulty)
displayedWord char[]
remainingAttempts int
validateGuess(char) void
hasSuccessfullyTerminated() boolean
displayedWord String
remainingAttempts int
guessedCharacters Set<Character>
gameOver boolean
Medium
Medium()
maximumAttempts int
Output
Output()
displayGameState(GameState) void
displayResult(boolean, GameState) void
displayWelcome() void
GameState
GameState(String, Difficulty)
validateGuess(char) void
Hard
Hard()
maximumAttempts int
Easy
Easy()
maximumAttempts int
Input
Input()
getValidGuess(GameState) char
Hangman
Hangman()
main(String[]) void
Game
Game(Difficulty)
run() void
initialiseGame() void
wordBank
difficulty
wordManager
outputHandler
inputHandler
gameState
«create»
«create»
«create»
«create»
1
*
1
1
1
1
1
1
1
1
1
1


User
Hangman
Game
FileWordManager
WordManager
GameState
Input
Output
Difficulty
main(args) - program is run
Display difficulty options
Select and enter difficulty
Create instance of chosen difficulty (Easy/Medium/Hard)
new Game(difficulty)
new FileWordManager()
loadWords()
Loaded word bank
Initialised
getRandomWord(difficulty)
wordToGuess
new GameState(wordToGuess, difficulty)
Initialised
loop
[Game loops until it's over (and the user wins/loses)]
displayGameState(gameState)
getDisplayedWord(), getRemainingAttempts()
Show game state to user
getValidGuess(gameState)
Prompt user for letter
Enter guess
Check guessedCharacters
Valid guess
validateGuess(guess)
Update displayedWord/remainingAttempts
State updated
displayResult(hasWon)
Show a win/lose message
Exit