Item	Occurance	Problem
1	Initial compile in BlueJ	Game.java uses unchecked or unsafe operations recompile with -Xlint: unchecked for details
2	General Commenting	Multiple methods are not well commented, the purpose or functionality of the code is not described
3	Game.createRooms()	sets the player's current room using a class variable of the Game class, despite the current room being a property of the player
4	class Game -> printWelcome()	System.out.println() used instead of \n where preceeding statement is also a print statement
5	class Game -> printWelcome()	overly used if conditionals; increases file size and difficult to read
6	class Game -> { printWelcome(), goRoom(), look() }	if conditional over currentRoom.exit to print direction is duplicated three times
7	class Game -> { processCommand(), quit() }	method takes an argument from the class command, but is located in class Game
8	class Game -> processCommand()	Readability - conditional checking fo unknown commands returns false
9	class Game -> printHelp()	when typing "help" into processCommand(), the correct string of commands in CommandWords.validCommands[] is not called by printHelp(); instead a string is outputted
10	Game.take(), .goRoom()	the user has entered take, go command without the argument for what, and asks user 'take what?', 'go where?'
11	Game.Game() { createRooms() }	generally the fields should be defined at the top of the constructor, followed by any functions. Plus the createRooms is an instantiation process and is only used once. There is no need to createRooms at any other time.
12	Game.printWelcome()	code is printed from a hard-coded string, which does not allow internalisation
13	Game.processCommand()	same problem with hard-coded stings as point 12
14	Game.printHelp()	same problem with hard-coded stings as point 12 also printHelp is almost a redundant function as all print functions could be one function or class that gets called with a variable to reference the desired output
15	Game.printHelp()	not printing all the commandWords as is a hardcoded string
16	Game { .processCommand(), .goRoom() }	using an if conditional statement that takes strings as inputs for comparison, hard-coded and slow
17	Game.take()	if statement used without an else condition where it could have been used instead of using another if conditional
18	Game.drop()	variable i is not understandable of what it means
19	Game.give()	command.hasThirdWord is always command.hasSecondWord
20	class Parser -> getCommand()	code is realloated as a new object for parsing, code duplication
21	class Parser -> getCommand()	fields wordX are not representing the data stored
22	class CommandWords method isCommand	does not check if input String is a String
23	CommandWords.isCommand()	for loops through an array
24	CommandWords	is inheriently part of Command class
25	Command.Command	built for three input words, need to allow extendability
26	Command.Command	this.thirdWord is copying secondWord for no apparent reason
27	command {.hasSecondWord .hasThirdWord}	provides no extra functionality as using command {.getSecondWord .getThirdWord}
28	Parser.Parser()	instantiatng commands with empty constructor in CommandWords
29	Parser.getCommand()	tokenizer is parsing an array of a string rather than scanning the string for the next word
30	Parser.getCommand()	only takes up to three words, it checks for less than three words, but not more than, plus no room for extendability
31	Parser.getCommand()	method is reading commands from user input, but is located in parser,

32	class Room	class Room has public fields that should not be accessible outside of its class
33	Room (fields)	Room fields are public and can be modified by anyone
34	Room (fields)	the exits are only north, east, south, west and does not allow extendability
35	Room.addItem()	no error checking on parameters to see if containing the correct type of variable
36	Room.character in Game.give()	Character takes default value null, but is used in Game.give() → throws NullPointerException
37	Room.addItem()	only allows room to have one item
38	Room.xItem()	not easy or efficient to identify an item from its description, plus a room could share the same description
39	Room.containsItem	only works if an item has a weight not equal to zero