# Getting Ready: The Chess Game

Understand the chess game problem and learn the questions to further simplify this problem.

**We'll cover the following**

* [Problem definition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Expectations-from-the-interviewee)
  + [Chess pieces](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-pieces)
  + [Gameplay](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Gameplay)
* [Design approach](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-approach)
* [Design patterns](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-patterns)

## Problem definition

**Chess** is a board game for two players that involves strategy and is played on a checkered board made up of 64 squares in an 8x8 grid. Each player starts with 16 pieces, including a king, queen, two rooks, two knights, two bishops, and eight pawns. The goal is to checkmate the opponent's king. This occurs when the king is in a position to be captured (check), and there is no way to move the king out of capture (checkmate).

Each piece has its own unique movements. The rook can move horizontally or vertically, the knight can move in an L-shape position, the bishop can move diagonally, the queen can move in any direction, and the king can move one square in any direction. Pawns have the most complex movement rules and can move forward one square but capture diagonally.

The game can also end in a draw if the king is not in checkmate, and it is not possible for either player to force a win. A draw happens if there are not enough pieces on the board to force a checkmate (except the two kings), the same position is repeated three times, or both the players agree to a draw.

**Note**: There are numerous variations of the game of chess that are played globally. In this design problem, we'll focus on creating a digital version of the two-player chess game that can be played online.

An online chess game

**1** of 11

## Expectations from the interviewee

There are several components in a chess game, each with specific constraints and requirements. The following provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail during the interview.

### Chess pieces

The chess pieces have their own specifications according to the rules of the game that need to be clarified by the interviewer. Therefore, you may ask the following questions:

1. How many chess pieces are there in the game?
2. What are the different chess pieces, and what are their respective moves?
3. Which piece is the weakest in chess?
4. Which piece is the strongest in chess?

### Gameplay

In chess, the two opposing teams try to achieve the state of checkmate to win the game, or they face some special cases like a stalemate that need to be clarified by the interviewer. You may ask the following questions regarding them:

1. Which player takes the first turn?
2. What are the rules of the game?
3. What is a checkmate?
4. How does a stalemate happen?
5. Can a player forfeit/resign from the game?

## Design approach

We'll design this chess game system using the bottom-up design approach. For this purpose, we will follow the steps below:

* Identify and design the smallest components first – the box and piece.
* Use these small components to design bigger components, for example, the chessboard and move.
* Repeat the steps above until we design the whole chess game.

## Design patterns

It is always a good practice to discuss the design patterns that the online chess game falls under during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns are used to design the online chess game:

* The Singleton design pattern
* The Command design pattern

Let's explore the requirements of the online Chess game in the next lesson.

Back

**Requirements for the Chess Game**

Learn about all requirements of the online Chess game.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Requirement-collection)
* [International chess rules](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#International-chess-rules)

In this lesson, we’ll list the requirements of the online chess game. This is a very crucial step since requirements define the scope of a problem, so getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label "Rn," where "R" is short for Requirement and "n" is a natural number.

**Requirement collection**

**R1:** The purpose of this system is to enable multiplayer in a game of chess via an online platform.

**R2:** The game will be played according to the official rules of an international chess game.

**R3:** Each player is randomly assigned the color—either black or white

**R4:** At the start of the game, each player will have eight pawns, two rooks, two bishops, two knights, one queen, and one king on the board.

**R5:** The player with the white pieces will make the first move.

**R6:** It is not possible for a player to retract or undo their move once it has been made.

**R7:** The system will keep a record of all moves made by both players.

**R8:** The game may end in a checkmate, forfeiture, stalemate (a draw), or resignation.

White pawn taking the first move

**International chess rules**

The following table represents the basic rules of chess:

**Rules for pieces**

|  |  |
| --- | --- |
| **Piece** | **Rules** |
| King | * It can move one step in any direction. * It cannot move to a box that might cause a check. |
| Queen | * It can move horizontally, vertically, or diagonally unless it is blocked by a piece of the opponent. * It cannot jump over the opponent's pieces. |
| Pawn | * It can move one box forward. * It is allowed to move two boxes forward if it is the first move by the player. * It can move one box diagonally to kill the opponent's pawn. |
| Bishop | * It can move only diagonally in any direction unless it is blocked by a piece of the opponent. * It cannot jump over the opponent's piece. |
| Rook | * It can move only horizontally or vertically unless it is blocked by a piece of the opponent. * It cannot jump over the opponent's piece. |
| Knight | * It can only move in an L-shape position by jumping two boxes horizontally or one box vertically. * It can jump over other pieces. |

The following table represents certain situations we might face while playing chess:

**Rules for situations**

|  |  |
| --- | --- |
| **Situation** | **Rules** |
| Checkmate | * This is when a player's king is in check (can be captured by the opponent's pieces) and there is no possible escape for the piece. * This situation decides the winner of the game. |
| Stalemate | * This is when a situation in which the player's king is not in check and no other move is possible. * It draws the match. |
| Forfeiture | * If a player does not show up for the game, then the player is considered to have forfeited. |
| Resignation | * If a player is at a position in the game where they understand that the stronger opponent will win in case of any move and decides to quit, then they have resigned from the game. |
| Castling | * A player moves their king two boxes towards the rook on the same row. * The rook is moved to the box the king passed over, which is next to the new position of the king. * The king and rook should be at their original positions and should not have been moved before. * No other piece should be between the king and the rook. |

We've identified our requirements for the problem. In the next lesson, we will define different use cases for the online chess game design.

# Use Case Diagram for the Chess Game

Learn how to define use cases and create the corresponding use case diagram for the chess game.

**We'll cover the following**

* [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Primary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Primary-actors)
  + [Secondary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Secondary-actors)
* [Use Cases](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-Cases)
  + [Player](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Player)
  + [Admin](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Admin)
* [Relationships](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationships)
  + [Include](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Include)
  + [Extend](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Extend)
  + [Generalization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Generalization)
  + [Associations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Associations)
* [Use case diagram](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-case-diagram)

Let’s build the use case diagram of the chess game and understand the relationship between its different components.

First, let’s define the different elements of our chess game, followed by the complete use case diagram of the system.

## System

Our system is a "Chess game."

## Actors

Here are the main actors of our chess game.

### Primary actors

* **Player:**This is the primary user and is responsible for playing the chess game. It can start a new game, make move or resign/forfeit a game.

### Secondary actors

* **Admin:** This can add, remove, or update a player's account and membership, view open games, and validate player moves.

## Use Cases

In this section, we'll define the use cases for the chess game. We have listed down the use cases according to their respective interactions with a particular actor.

**Note:**You will see some use cases occurring multiple times because they are shared among different actors in the system.

### Player

* **Create/Update account:**To create a new account to play a chess game or to update account information of an existing account
* **Login/Logout:** To log in to or log out from an account
* **View open games:** View available games that are waiting for players to join
* **Join a game:** Join an open game
* **Create a new game:** To create a new game to start playing
* **Make move:** To make a move in the game
* **Resign or forfeit a game:** To resign or forfeit a game so that it ends

### Admin

* **Block/unblock member:**To block or unblock a member from playing the chess game
* **Cancel/Update membership:** To cancel the membership or to update the membership of an account
* **Add/modify member:** To add a new member or update member information
* **Login/Logout:** To log in to or log out from an account
* **View open games:** View available games that are waiting for players to join
* **Validate Moves:**To validate player move
* **Declare results:**To declare the result of the game when the game is over

## Relationships

This section describes the relationships between and among actors and their use cases.

### Include

* The “Make move” has an include relationship with the “Validate move” because the admin has to validate if the move was as per the rules set.
* The “Resign or forfeit a game” use case has an include relationshop with the “Declare result” use case because the game will be over, and results will be declared when a player either resigns or forfeit from the game.

### Extend

* The “Block/unblock member” has an extend relationship with the “Cancel/Update membership” since when the admin unblocks a member, there is a chance that their membership will be updated. When the admin blocks a member, its membership might be canceled.
* The “Validate move” use case has an extend relationshop with the “Declare result” use case as there is a chance that the game will be over and results will be declared when a player makes a validate move and checkmate another player.

### Generalization

The “Make move” has a generalization relationship with the “Play pawn”, “Play bishop”, “Play king”, “Play queen”, “Play knight”, and “Play rook”, since a player can make any of these six moves.

### Associations

The table below shows the association relationship between actors and their use cases.

|  |  |
| --- | --- |
| **Player** | **Admin** |
| Create/Update account | Block/unblock member |
| Join a game | Cancel/Update membership |
| Create a new game | Add/modify member |
| Make move | Login/Logout |
| Resign or forfeit a game | Validate Moves |
| View open games | View open games |
| Login/Logout | Declare results |

## Use case diagram

Here is the use case diagram of the chess game:

# A screenshot of a computer Description automatically generatedClass Diagram for the Chess Game

Learn to create a class diagram for the chess game using the bottom-up approach.

**We'll cover the following**

* [Components of Chess](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-Chess)
  + [Box](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Box)
  + [Chessboard](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chessboard)
  + [Piece](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Piece)
  + [Move](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Move)
  + [Account, Player, and Admin](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Account,-Player,-and-Admin)
  + [Chess move controller](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-move-controller)
  + [Chess game view](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-game-view)
  + [Chess game](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-game)
  + [Enumerations and custom data types](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations-and-custom-data-types)
* [Relationship between the classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationship-between-the-classes)
  + [Association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Association)
  + [Aggregation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Aggregation)
  + [Composition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Composition)
  + [Inheritance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inheritance)
* [Class diagram for the Chess game](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Class-diagram-for-the-Chess-game)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)

We’ll create the class diagram for the chess game. In the class diagram, we will first design the classes and then identify the relationship between classes according to the requirements for the chess game problem.

## Components of Chess

As mentioned earlier, we'll follow the bottom-up approach to designing a class diagram for the chess game.

### Box

A Box is a specific position/block on the 8x8 chessboard which is defined by row x and column y, respectively.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a game

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

The following classes demonstrate an inheritance relationship:

* Both, Admin and Player extend the Account class.
* The King, Queen, Knight, Bishop, Rook, and Pawn classes extend the Piece class.

**Note**: We have already discussed the inheritance relationship between classes in the component section above one by one.

## Class diagram for the Chess game

Here’s the complete class diagram for our chess game:

A diagram of a game

Description automatically generated

The following design patterns have been used in the class diagram:

* **Singleton design pattern:** This pattern ensures the existence of a single instance of the chessboard at a given moment due to the shared nature of the chessboard as a resource. Multiple instances can cause the game state to become inconsistent.
* **Command design pattern:** This pattern is used to encapsulate the move logic for each chess piece. Each chess piece has its own implementation of the move command, which allows it to move according to the rules defined for it. For example, the knight moves in an L-shape pattern, or the rook can move only horizontally or vertically on any number of boxes.

The following design patterns can also be used to design chess:

* The Iterator design pattern would enable the game to move sequentially by allowing the pieces to behave in a uniform manner where the user does not need to know the specifications or underlying logic behind the moves of the pieces.
* The State design pattern ensures the encapsulation of the state logic of each piece, since all the chess pieces have their own respective implementations of checkmate states which makes them behave differently from each other depending on the situation.
* The Observer design pattern enables the chess pieces to act as observers where the chessboard is the subject. As soon as the state of the board changes, the pieces are notified to adapt to the changes accordingly. This decouples the pieces from the chessboard.

We have completed the class diagram of the chess game according to the requirements. Now, let's design its activity diagram in the next lesson.

Back

# Activity Diagram for the Chess Game

Let's create an activity diagram for the chess game.

**We'll cover the following**

* [Chess game](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-game)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)

An activity diagram is a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for our Chess game. In this lesson, we will create an activity diagram to show how to play chess.

## Chess game

The following states and actions will be involved in this activity diagram.

### States

**Initial state:**The player makes the first move.

**Final state:**Any of the game-over condition are met.

### Actions

The player initiates a new game. The board appears, and the player takes a turn. The system validates the move, then system checks if any of the game-over conditions are applicable. If the game is not over, the other player takes his turn, and the game continues until any of the game-over conditions are met.

Based on the order above, the activity diagram of chess game is given below.

A diagram of a chess game

Description automatically generated

We've looked at the activity diagram of our chess game. In the next lesson, we will present the code for our designed classes in some of the most popular languages.

# Code for the Chess Game

Write the object-oriented code to implement the design of the chess game problem.

**We'll cover the following**

* [Chess game classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-game-classes)
  + [Enumerations and custom data type](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations-and-custom-data-type)
  + [Box and chessboard](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Box-and-chessboard)
  + [Piece](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Piece)
  + [Move](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Move)
  + [Account, player, and admin](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Account,-player,-and-admin)
  + [Chess move controller and the game view](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-move-controller-and-the-game-view)
  + [Chess game](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Chess-game)
* [Wrapping up](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Wrapping-up)

We've covered different aspects of the chess game and observed the attributes attached to the problem using various UML diagrams. Let's now explore the more practical side of things where we will work on implementing the chess game using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the chess game:

* Java
* C#
* Python
* C++
* JavaScript

## Chess game classes

In this section, we will provide the skeleton code of the classes designed in the class diagram lesson.

**Note:** For simplicity, we are not defining getter and setter functions. The reader can assume that all class attributes are private and accessed through their respective public getter methods and modified only through their public method functions.

### Enumerations and custom data type

The following code provides the definition of the enumeration and custom data type used in the chess game.

GameStatus: This enumeration keeps track of the active status of the player and the game, i.e, who wins and whether or not the game is a draw.

AccountStatus: We need to create an enumeration to keep track of the status of the account – whether it is active, canceled, closed, blocked, or none.

The Person class is used as a custom data type. The implementation of the Person class can be found below:

**Note:** JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

Java

// Enumerations

enum GameStatus {

Active,

BlackWin,

WhiteWin,

Forfeit,

Stalemate,

Resignation

}

enum AccountStatus {

ACTIVE,

CLOSED,

CANCELED,

BLACKLISTED,

NONE

}

// Custom Person data type class

public class Person {

private String name;

private String streetAddress;

private String city;

private String state;

private int zipCode;

private String country;

}

### Box and chessboard

The Box class holds the piece where the Chessboard contains the boxes and has the functionality of updating or resetting the board. The definitions of these classes are provided below:

public class Box {

private Piece piece;

private int x;

private int y;

}

public class Chessboard {

private Box[][] boxes;

private Date creationDate;

public List<Piece> getPieces()

public void resetBoard()

public void updateBoard()

}

### Piece

Piece is an abstract class that is extended by King, Queen, Knight, Bishop, Rook and Pawn. These derived classes override the canMove() function of Piece. The definitions of these classes are provided below:

public abstract class Piece {

private boolean killed = false;

private boolean white = false;

public boolean isWhite();

public boolean isKilled();

public abstract boolean canMove(Chessboard board, Box start, Box end);

}

public class King extends Piece {

private boolean castlingDone = false;

@Override

public boolean canMove(Board board, Box start, Box end) {

// definition

}

}

public class Queen extends Piece {

@Override

public boolean canMove(Board board, Box start, Box end) {

// definition

}

}

public class Knight extends Piece {

@Override

public boolean canMove(Board board, Box start, Box end) {

// definition

}

}

public class Bishop extends Piece {

@Override

public boolean canMove(Board board, Box start, Box end) {

// definition

}

}

public class Rook extends Piece {

@Override

public boolean canMove(Board board, Box start, Box end) {

// definition

}

}

public class Pawn extends Piece {

@Override

public boolean canMove(Board board, Box start, Box end) {

// definition

}

}

A screenshot of a computer program

Description automatically generated

### Account, player, and admin

The Account class is extended by the Player and Admin classes.

* The Player class records the player's information by storing the Person object, along with the chosen color, i.e., – whether or not the player is playing with white pieces.
* The Admin class decides whether or not the user is blocked.

The definitions of these classes are provided below:

public class Account {

private int id;

private String password;

private AccountStatus status;

public boolean resetPassword();

}

public class Player extends Account {

private Person person;

private boolean whiteSide = false;

private int totalGamesPlayed;

public boolean isWhiteSide();

public boolean isChecked();

}

public Admin extends Account {

public boolean blockUser();

}

A screenshot of a computer

Description automatically generated

public class ChessGame {

private Player[] players;

private Chessboard board;

private Player currentTurn;

private GameStatus status;

private List<Move> movesPlayed;

public boolean isOver();

public boolean playerMove(Player player, int startX, int startY, int endX, int endY) {

/\* 1. start box

2. end box

3. move

4. call makeMove() method

\*/

}

private boolean makeMove(Move move, Player player) {

/\* 1. Validation of source piece

2. Check whether or not the color ofthe piece is white

3. Check if it is a valid move or not

4. Check whether it is a castling move or not

5. Store the move

\*/

}

}

## Wrapping up

We've explored the complete design of the chess game in this chapter. We've looked at how a basic chess game can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.