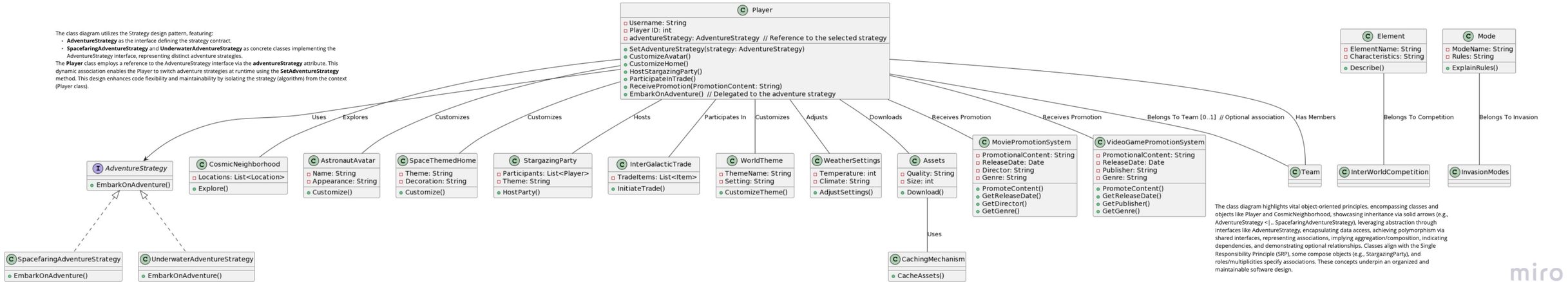
The class diagram utilizes the Strategy design pattern, featuring:
AdventureStrategy as the interface defining the strategy contract.
SpacefaringAdventureStrategy and UnderwaterAdventureStrategy as concrete classes implementing the AdventureStrategy interface, representing distinct adventure strategies.
The Player class employs a reference to the AdventureStrategy interface via the adventureStrategy attribute. This dynamic association enables the Player to switch adventure strategies at runtime using the SetAdventureStrategy method. This design enhances code flexibility and maintainability by isolating the strategy (algorithm) from the context (Player class).
Player
Username: String
Player ID: int
adventureStrategy: AdventureStrategy // Reference to the selected strategy
SetAdventureStrategy(strategy: AdventureStrategy)
CustomizeAvatar()
CustomizeHome()
HostStargazingParty()
ParticipateInTrade()
ReceivePromotion(PromotionContent: String)
EmbarkOnAdventure() // Delegated to the adventure strategy
Uses
Explores
Customizes
Customizes
Hosts
Participates In
Customizes
Adjusts
Downloads
Receives Promotion
Receives Promotion
Belongs To Team [0..1] // Optional association
Has Members
Element
ElementName: String
Characteristics: String
Describe()
Mode
ModeName: String
Rules: String
ExplainRules()
Belongs To Competition
Belongs To Invasion
AdventureStrategy
EmbarkOnAdventure()
CosmicNeighborhood
Locations: List<Location>
Explore()
AstronautAvatar
Name: String
Appearance: String
Customize()
SpaceThemedHome
Theme: String
Decoration: String
Customize()
StargazingParty
Participants: List<Player>
Theme: String
HostParty()
InterGalacticTrade
TradeItems: List<Item>
InitiateTrade()
WorldTheme
ThemeName: String
Setting: String
CustomizeTheme()
WeatherSettings
Temperature: int
Climate: String
AdjustSettings()
Assets
Quality: String
Size: int
Download()
MoviePromotionSystem
PromotionalContent: String
ReleaseDate: Date
Director: String
Genre: String
PromoteContent()
GetReleaseDate()
GetDirector()
GetGenre()
VideoGamePromotionSystem
PromotionalContent: String
ReleaseDate: Date
Publisher: String
Genre: String
PromoteContent()
GetReleaseDate()
GetPublisher()
GetGenre()
Team
InterWorldCompetition
InvasionModes
SpacefaringAdventureStrategy
EmbarkOnAdventure()
UnderwaterAdventureStrategy
EmbarkOnAdventure()
Uses
CachingMechanism
CacheAssets()
The class diagram highlights vital object-oriented principles, encompassing classes and objects like Player and CosmicNeighborhood, showcasing inheritance via solid arrows (e.g., AdventureStrategy <|.. SpacefaringAdventureStrategy), leveraging abstraction through interfaces like AdventureStrategy, encapsulating data access, achieving polymorphism via shared interfaces, representing associations, implying aggregation/composition, indicating dependencies, and demonstrating optional relationships. Classes align with the Single Responsibility Principle (SRP), some compose objects (e.g., StargazingParty), and roles/multiplicities specify associations. These concepts underpin an organized and maintainable software design.
miro