Software Design – Autumn 2014

Course code: TI00AA56
Slide set № 5

24.9.2014
Olli Hämäläinen

Case
“African Star”

Requirement Specification

The game shall be implemented as a computer application with a graphical user interface. The game is played by two or more players. Each player has a playing piece, which all look different. The pieces are moved on the playing board. When the game starts, all pieces are in the starting place and each player recieves a sum of game money. The game has a set of places, on which the pieces can be positioned. The places are either ordinary places (black spots) or special places (red spots). Some pairs of these places are connected by a connection. Graphically, the connections form paths on the board. There are flight connections for some pairs of places.

Requirement Specification

The players play in turns. The game has a die, which shows how many steps each playing piece is to be moved in each turn. With each step, the player's piece is moved from one place to another place using a connection on condition that these places are connected. A player may use a flight connection by paying one unit of game money, assuming he is at one end of the flight connection when his/her turn comes.

Req.Spec. cont'd

At the start of the game each special place has a card face down. When a player arrives to a special place, he/she may pay one unit of game money to take the card (which can only be taken once). Alternatively, the player may stay on a special place, and instead of moving, throw the die for trying to take the card: die values 4,5, and 6 allow the player to get the card.

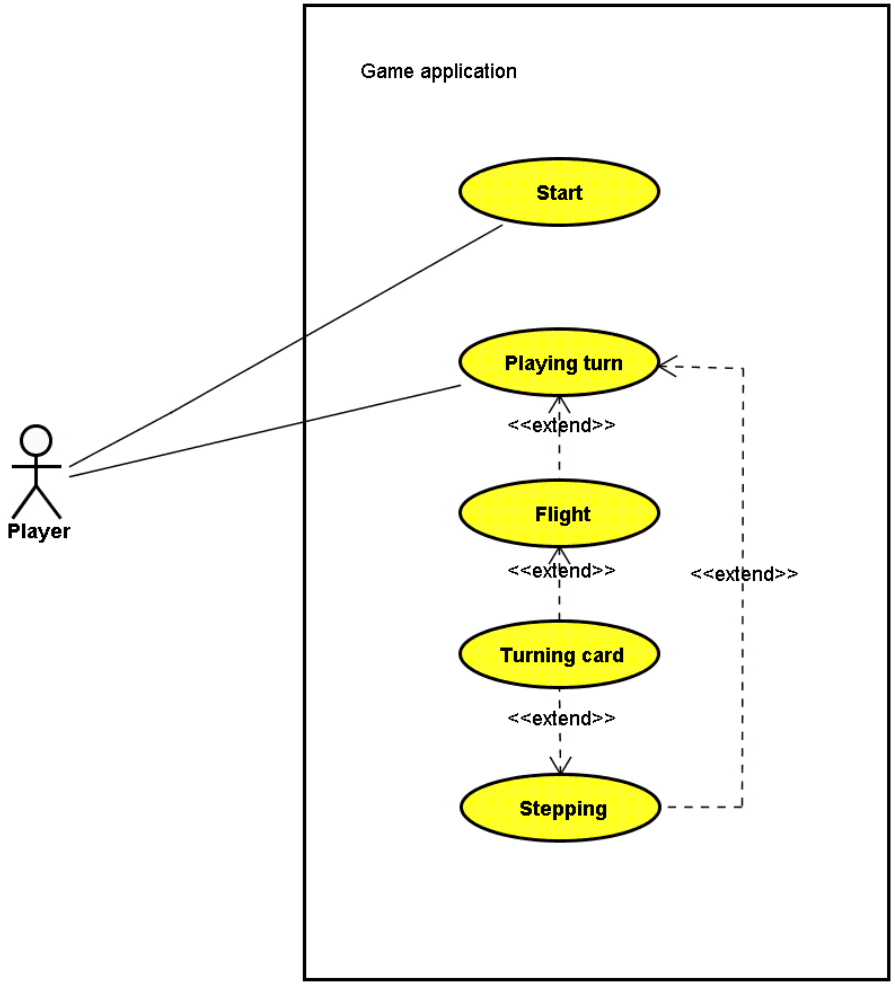
Req.Spec. cont'd

The card may be a bandit, meaning the player loses all money, a jewel with a value, meaning that the player receives money for that value, a horse shoe, or a treasure (the African Star diamond). The card may also be empty. The treasure may not be sold, but if another player arrives to a place which has a player with the treasure, the arriving player gets the treasure. When a player with the treasure arrives to the starting place, that player wins the game and the game ends. After the treasure has been found any player with a horse shoe can win by arriving at the starting place before the player with the treasure.

Req.Spec. cont'd

The game boards, places, pieces, labels, connections, etc. each have a graphical representation in the user interface, but that representation is not specified here.

Use Case Diagram of ”African Star”



Use Case ”Start”

Name: Start Version: 1.0

Summary: The game is initialised

Frequency: Once per every game played

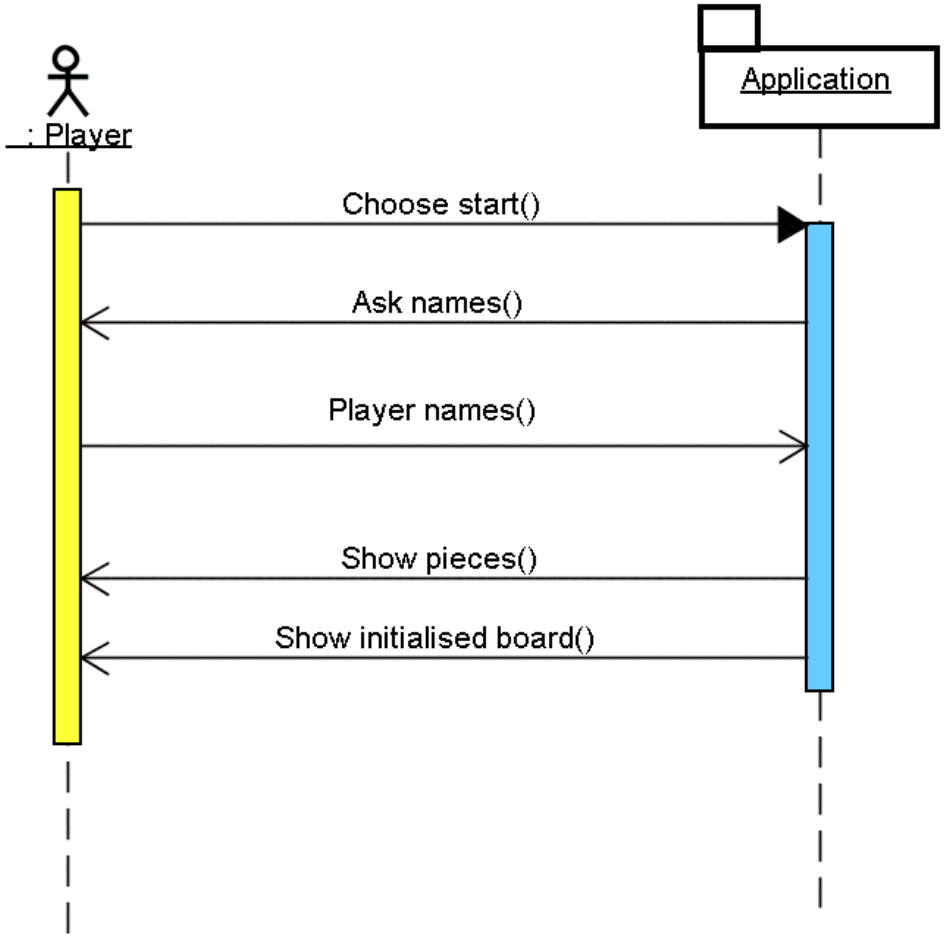
Actors: Players

Preconditions:

Descriptions: The game application shows a dialogue, where all players’ names are typed in. The game application randomly chooses the game pieces for the players and the order in which players take turns. The game starting position is initialised by putting the pieces in the start place and randomly choosing the cards for the special places.

Postconditions:
Exceptions:

Sequence Diagram: ”Start”



Use Case ”Playing turn ”

Name: Playing turn Version: 1.0

Summary: Player chooses how to proceed: flight or other

Frequency: Once for every player during a playin round

Actors: Player

Preconditions: It is the player’s turn to proceed.

Descriptions:

If the player is in a special place and has money,

the player is asked if she/he wishes to fly or not.

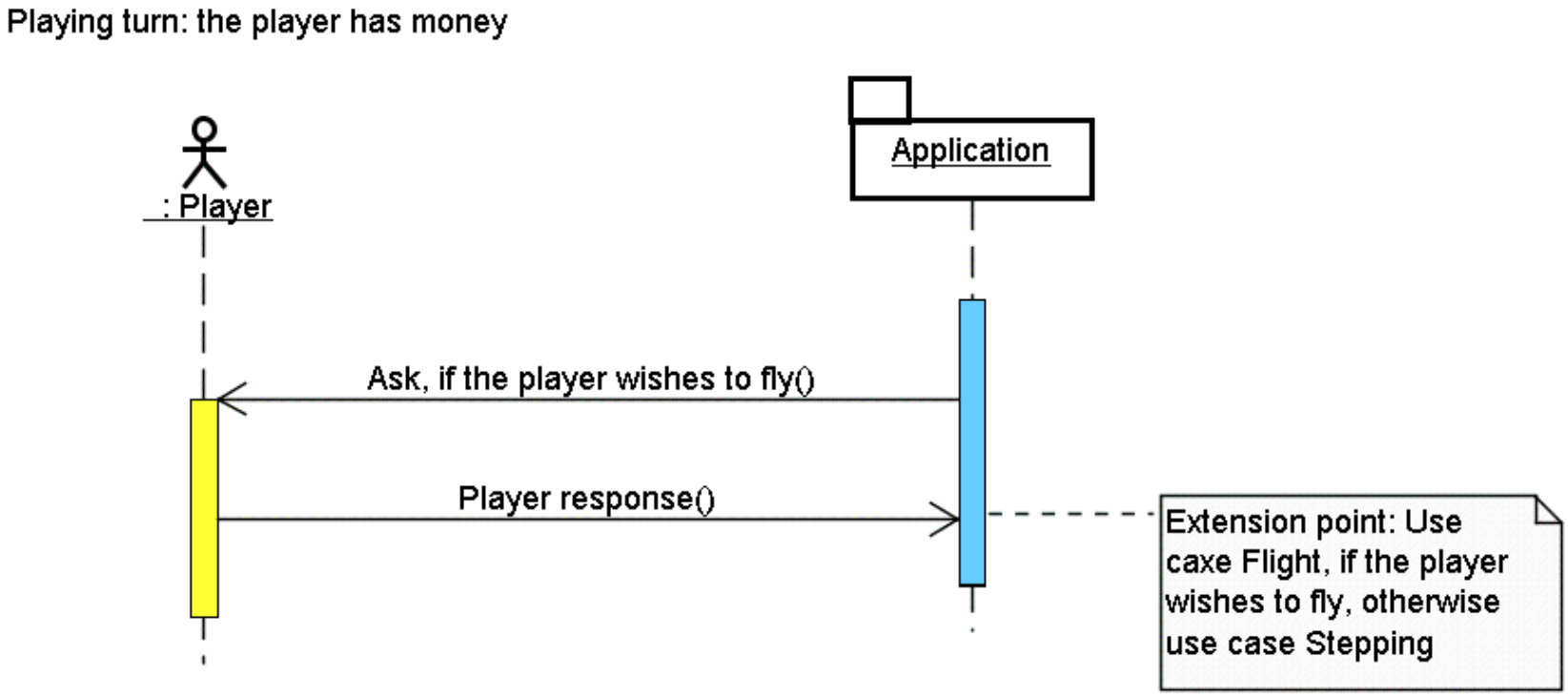
The use case is then extended by use case ”Flight” or ”Stepping” accordingly.

If the player has no money or is not in a special place, the use case is extended by use case ”Stepping”.

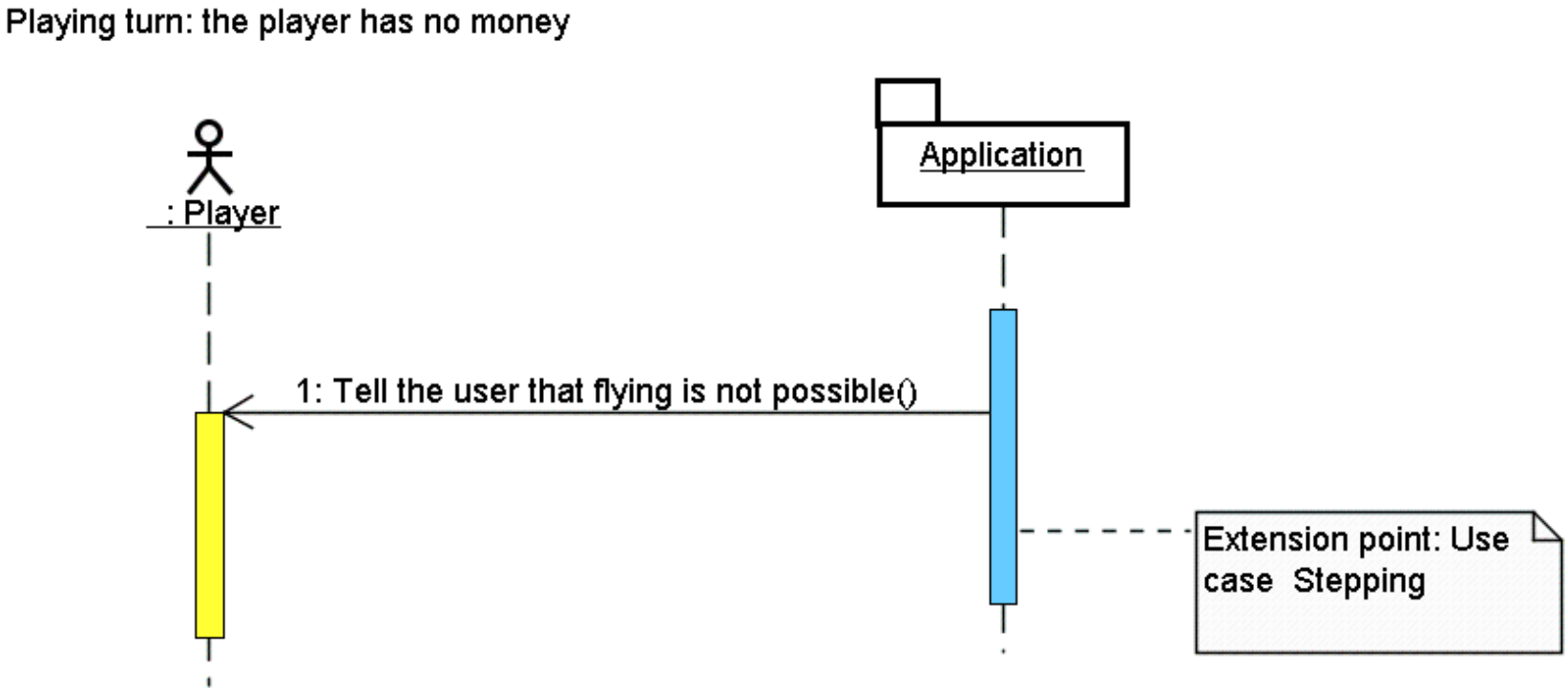
Postconditions:

Exceptions:

Sequence Diagram: ”Playing turn” (1)



Sequence Diagram: ”Playing turn” (2)



Use Case ”Turning Card”

Name: Turning Card Version: 1.0

Summary: Player turns a card

Frequency: More seldom than once every round

Actors: Player

Preconditions: It is the player’s turn, the player has money or thrown 4,5,or 6, and the player is in a special place with a card that has not been taken.

Descriptions: The player chooses to turn the card.

One unit of money is taken from the player, if the player has any.

If the card is empty, nothing happens.

If the card is a jewel, the player’s funds are increased with the respective amount of money.

If the card is a bandit, the player’s funds are set to zero.

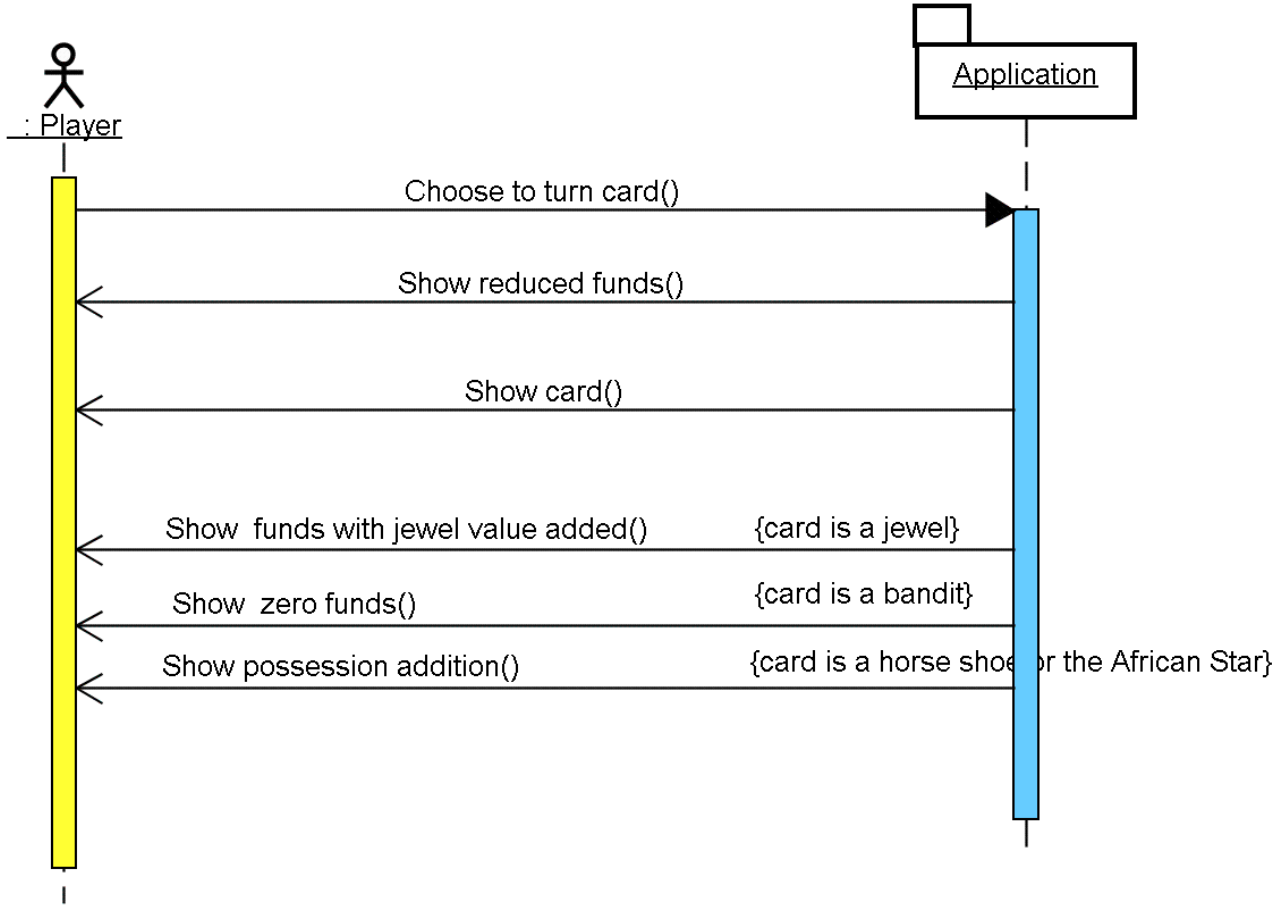
If the card is a horse shoe, the possession of a horse shoe is registered.

If the card is the African Star, the possession is registered.

Postconditions:

Exceptions:

Sequence Diagram: ”Turning card”



Use Case ”Flight”

Name: Flight Version: 1.0

Summary: Player flies from one place to another

Frequency: Probably not as often as once every round

Actors: Player

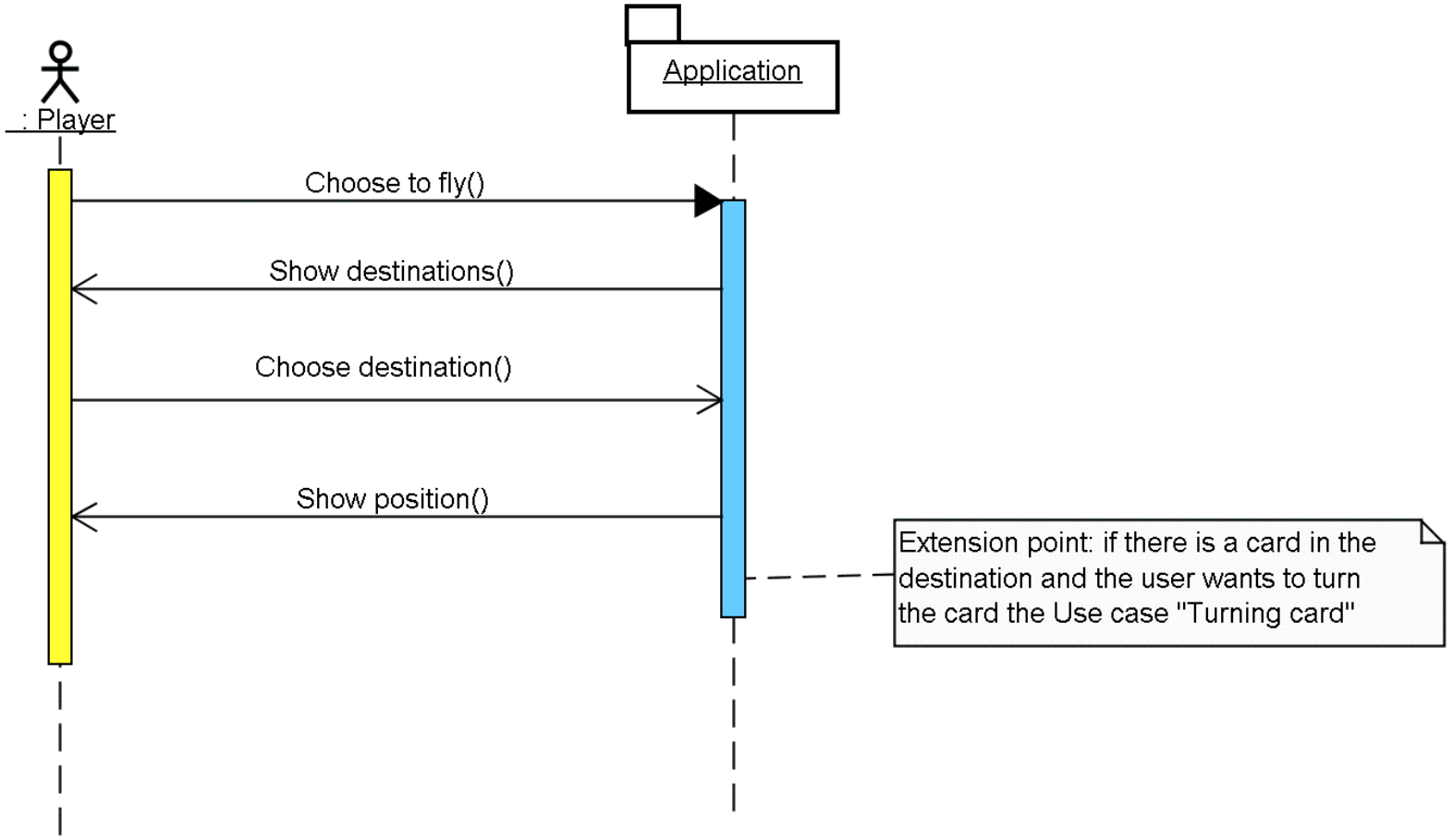
Preconditions: It is the player’s turn, the player has money to fly, and the player is in a place, where there are flight connections.

Descriptions: The player chooses to fly. The player is shown the possible destinations, of which the player chooses one. The game application moves player’s piece to the chosen destination, and his funds are decremented by one unit. If there is a card in the destination, the user has money, and the user wants to turn the card, this use case is extended by use case ”Turning card”.

Postconditions:

Exceptions:

Sequence Diagram for ”Flight”



Use Case ”Stepping”

Name: Stepping Version: 1.0

Summary: Player uses a playing turn to move according to the die or tries to get a number that allows to turn a card.

Frequency: Most players do this on most rounds

Actors: Player

Preconditions: It is the player’s turn. The player has decided not to fly orhas no money.

Description: The player chooses between the two alternatives: 1. proceed stepping by land or 2. try to get a result allowing to turn the card in the place where the player is located. The player throws the die.

If the player has chosen alternative 1, the game application shows the result and the possible places where to move to. The player chooses one destination. The game application moves player’s piece to the new place. If there is a card in the destination and the player has money, the player is asked if she/he wishes to turn the card. If the player wishes to turn, this use case is extended by use case ”Turning card”.

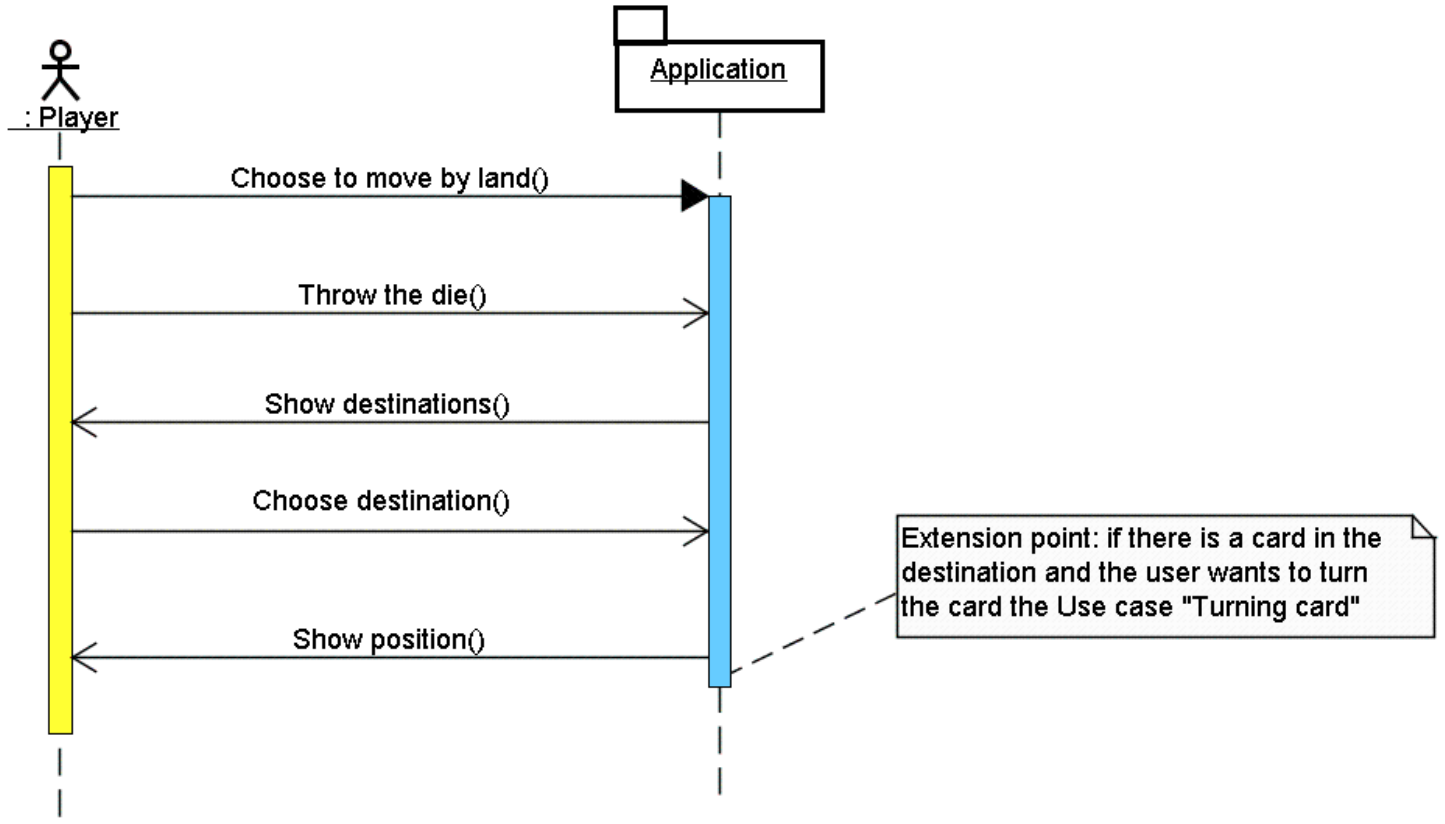
If the player has chosen alternative 2, and the result of throwing the die is 4, 5, or 6, this use case is extended by use case ”Turning card”.

Postconditions:

Exceptions:

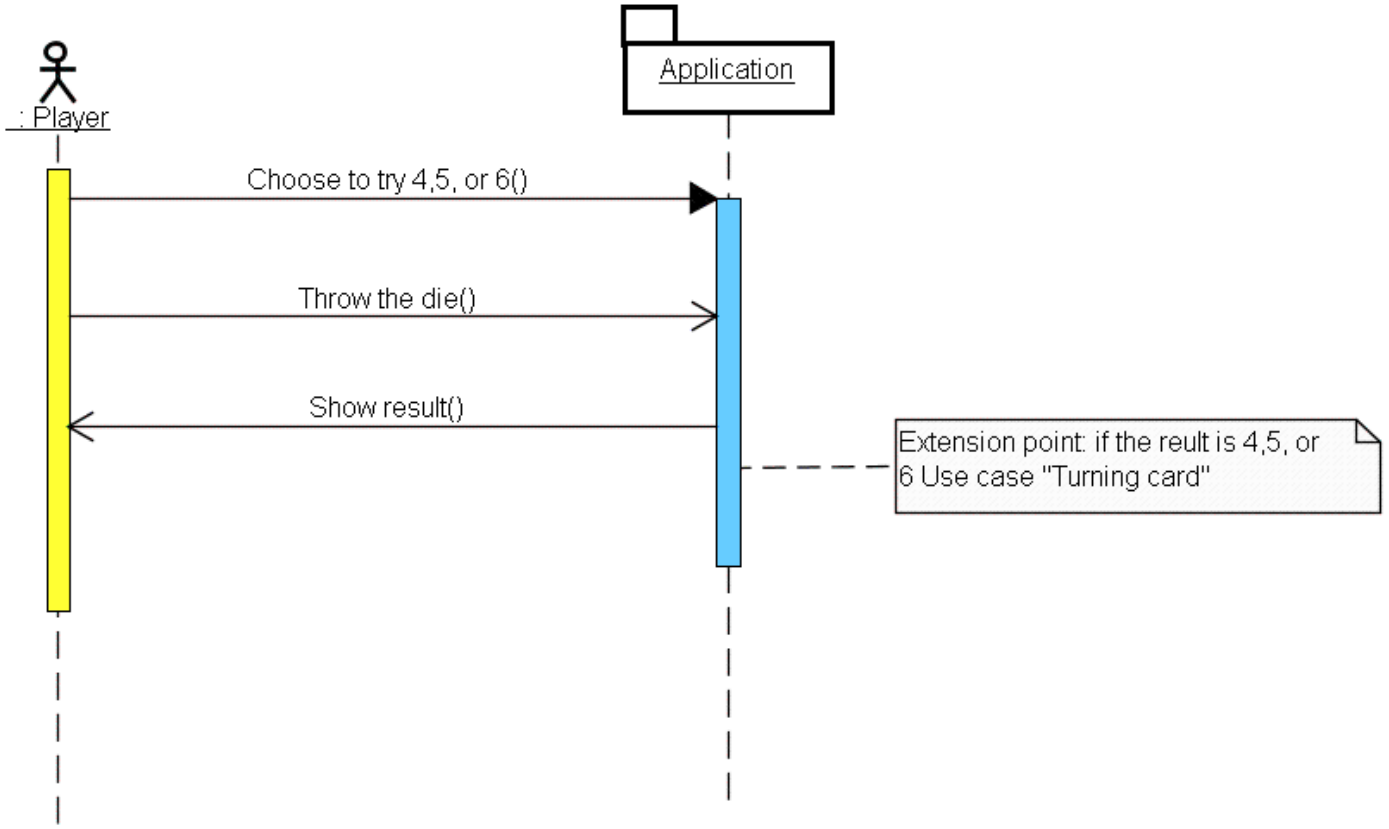
Sequence Diagram for ”Stepping” (1)

Altenative 1: proceed by land



Sequence Diagram for ”Stepping” (2)

Altenative 2: try to get high score



Analysis model: Classes

Text analysis: go through the text, and pick nouns

Nouns in Req. Specification (1)

The game shall be implemented as a computer application with a graphical user interface. The game is played by two or more players. Each player has a playing piece, which all look different. The pieces are moved on the playing board. When the game starts, all pieces are in the starting place and each player recieves a sum of game money. The game has a set of places, on which the pieces can be positioned. The places are either ordinary places (black spots) or special places (red spots). Some pairs of these places are connected by a connection. Graphically, the connections form paths on the board.

Nouns in Req. Specification (2)

There are flight connections for some pairs of places.

The players play in turns. The game has a die, which shows how many steps each playing piece is to be moved in each turn. With each step, the player's piece is moved from one place to another place using a connection on condition that these places are connected. A player may use a flight connection by paying one unit of game money, assuming he is at one end of the flight connection when his/her turn comes.

Nouns in Req. Specification (3)

At the start of the game each special place has a card face down. When a player arrives to a special place, he/she may pay one unit of game money to take the card (which can only be taken once). Alternatively, the player may stay on a special place, and instead of moving, throw the die for trying to take the card: die values 4,5, and 6 allow the player to get the card.

Nouns in Req. Specification (4)

The card may be a bandit, meaning the player loses all money, a jewel with a value, meaning that the player receives money for that value, a horse shoe, or a treasure (the African Star diamond). The card may also be empty. The treasure may not be sold, but if another player arrives to a place which has a player with the treasure, the arriving player gets the treasure. When a player with the treasure arrives to the starting place, that player wins the game and the game ends. After the treasure has been found any player with a horse shoe can win by arriving at the starting place before the player with the treasure.

Nouns in Req. Specification (5)

The game boards, places, pieces, labels, connections, etc. each have a graphical representation in the user interface, but that representation is not specified here.

Nouns in the Use Case ”Start”

Descriptions: The game application shows a dialogue, where all players’ names are typed in. The game application randomly chooses the game pieces for the players and the order in which players take turns. The game starting position is initialised by putting the pieces in the start place and randomly choosing the cards for the special places.

Nouns in ”Playing turn ”

Descriptions:

If the player is in a special place and has money,

the player is asked if she/he wishes to fly or not. The use case is then extended by use case ”Flight” or ”Stepping” accordingly.

If the player has no money or is not in a special place, the use case is extended by use case ”Stepping”.

Nouns in ”Turning Card”

Descriptions: The player chooses to turn the card.

One unit of money is taken from the player, if the player has any.

If the card is empty, nothing happens.

If the card is a jewel, the player’s funds are increased with the respective amount of money.

If the card is a bandit, the player’s funds are set to zero.

If the card is a horse shoe, the possession of a horse shoe is registered.

If the card is the African Star, the possession is registered.

Nouns in ”Flight”

Descriptions: The player chooses to fly. The player is shown the possible destinations, of which the player chooses one. The game application moves player’s piece to the chosen destination, and his funds are decremented by one unit. If there is a card in the destination, the user has money, and the user wants to turn the card, this use case is extended by use case ”Turning card”.

Nouns in ”Stepping”

Description: The player chooses between the two alternatives: 1. proceed stepping by land or 2. try to get a result allowing to turn the card in the place where the player is located. The player throws the die.

If the player has chosen alternative 1, the game application shows the result and the possible places where to move to. The player chooses one destination. The game application moves player’s piece to the new place. If there is a card in the destination and the player has money, the player is asked if she/he wishes to turn the card. If the player wishes to turn, this use case is extended by use case ”Turning card”.

If the player has chosen alternative 2, and the result of throwing the die is 4, 5, or 6, this use case is extended by use case ”Turning card”.

Potential classes

game	set (of places)	die
computer application	place	step
user interface	(red/black) spot	(on) condition
player	connection	unit (of money)
(playing) piece	path	card
(playing) board	flight (connection)	face
starting place	pair (of places)	(die) values
sum (of money)	turn	bandit

Potential classes, continued

jewel	label	”Turning card”
horse shoe	dialogue	amount (of money)
treasure	(player’s) name	fund(s)
”African star”	(turn) order	zero
diamond	(starting) position	possession (of smth.)
starting place	use case	destination
representation	”Flight”	user
special place	”Stepping”	alternative

Potential classes, continued

result		
land		
representation		

Reasons for rejecting potential classes

- irrelevant
- essentially same as some other class
- a likely attribute or a value
- a likely method
- a control-related concept
- a role of another class
- an association between classes
- vague
- implementation-specific

Accepting a class

- Typically checked by examining an object of the class.
- Retained information – the system needs some information about object to function.
- Needed services – the object has operations, which change its attribute values.
- Multiple attributes (a single-attribute class seems like a ”minor” thing at this stage).
- Common attributes (a set of attributes, which apply to all objects of the class.
- Common operations (like with attributes above)
- Essential requirements – external entities, which consume or produce information will almost always be modelled as objects in the analysis model.

Nouns rejected as classes (1)

- computer, computer application (irrelevant)
- board (represents the same thing as path) ->
  in fact, we choose to use ”map” for both of them
- amount, amount of money (a likely attribute)
- game money, dice value, result, zero, funds (likely values for attributes: a class is probably not needed)
- step (a likely operation)
- turn, turn order (control-related)
- starting place (a role for place)
- representation (irrelevant)
- jewel value (a likely attribute)
- pair of places, set of places (represents the same thing as connection)

Nouns rejected as classes (2)

- unit (irrelevant)
- bandit, jewel, horse shoe (values of an attribute) – treasure is a bit of a question mark!
- (red/black) spot (the same as place)
- (on) condition (irrelevant)
- use case, ”Flight”, ”Stepping”, ”Turning card” – irrelevant/control
- destination – same as place or special place
- user – same as player
- alternative, dialogue – control
- user interface, face down, starting position, – outside the scope (user interface related)

Nouns rejected as classes (3)

- player's name – an attribute
- diamond, "African star" – same as treasure
- possession – related to attribute values

Trimming classes

game	set (of places)	die
computer application	place	step
user interface	(red/black) spot	(on) condition
player	connection	unit (of money)
(playing) piece	path	card
(playing) board	flight (connection)	face
starting place	pair (of places)	(die) values
sum (of money)	turn	bandit

Trimming classes, continued

jewel	label	”Turning card”
horse shoe	dialogue	amount (of money)
treasure	(player’s) name	fund(s)
”African star”	(turn) order	zero
diamond	(starting) position	possession (of smth.)
starting place	use case	destination
representation	”Flight”	user
special place	”Stepping”	alternative

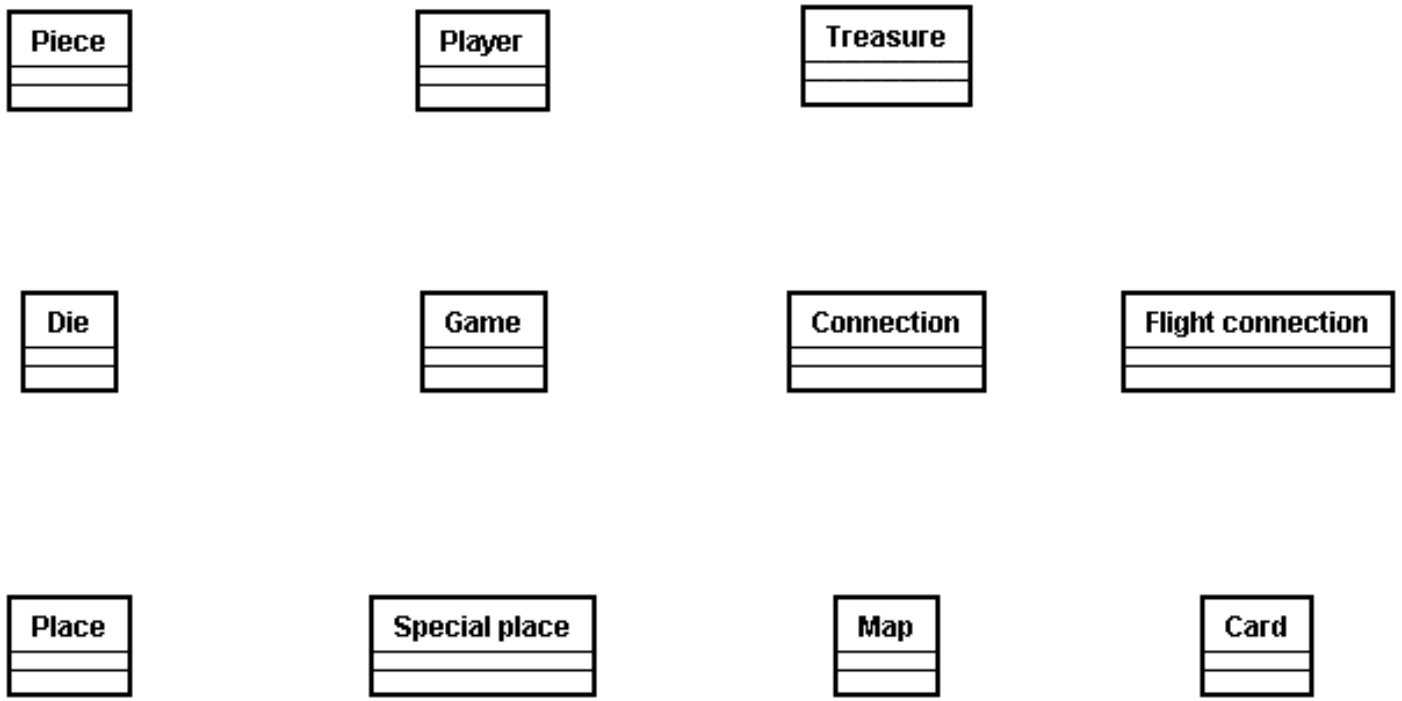
Trimming classes, continued

result		
land		
representation		
money		

Nouns accepted as classes

card	map	special place
connection	piece	treasure
flight connection	game	player
die	place	

Initial Class Diagram



Start of the initial model glossary

Bandit	A possible symbol to appear on a card. If a player takes a card, and the card is found to have a bandit, the player loses all of his/her funds.
Die	Randomly returns one of values 1, 2, 3, 4, 5 or 6. The return value is used to showing how many steps a player may advance or, alternatively, if a player gets to take card for free, in which case at least 4 is required.
Game piece	Each player has a game piece, which is used to show the position of the player. Every game piece looks different.
…	…

Task list

- Once the use case sequence diagrams have been produced, they are analysed for forming a task list.
- Each arrow from the user to the application represents a service requested from the application.
- Each arrow from the application to the user represents an output from the application.

Task List for The Game Application

- Initialise game
- Show board, piece positions, funds, and game status
- Show die result
- Show a set of destination places
- Choose to use the dice
- Choose to fly
- Choose to start a game
- Choose to open a card
- Choose a destination from a set of places
- Ask player names
- Input player names

Associations

- Associations are relationships, references, and dependencies between classes.
- In practice, they can be found in sentences, which represent relationships between classes.
- Examples of types of associations:
  - location
  - control
  - communication
  - inclusion
  - &c.

Verbs in Req. Specification (1)

The game shall be implemented as a computer application with a graphical user interface. The game is played by two or more players. Each player has a playing piece, which all look different. The pieces are moved on the playing board. When the game starts, all pieces are in the starting place and each player recieves a sum of game money. The game has a set of places, on which the pieces can be positioned. The places are either ordinary places (black spots) or special places (red spots). Some pairs of these places are connected by a connection. Graphically, the connections form paths on the board.

Verbs in Req. Specification (2)

There are flight connections for some pairs of places.

The players play in turns. The game has a die, which shows how many steps each playing piece is to be moved in each turn. With each step, the player's piece is moved from one place to another place using a connection on condition that these places are connected. A player may use a flight connection by paying one unit of game money, assuming he is at one end of the flight connection when his/her turn comes.

Verbs in Req. Specification (3)

At the start of the game each special place has a card face down. When a player arrives to a special place, he/she may pay one unit of game money to take the card (which can only be taken once). Alternatively, the player may stay on a special place, and instead of moving, throw the die for trying to take the card: die values 4,5, and 6 allow the player to get the card.

Verbs in Req. Specification (4)

The card may be a bandit, meaning the player loses all money, a jewel with a value, meaning that the player receives money for that value, a horse shoe, or a treasure (the African Star diamond). The card may also be empty. The treasure may not be sold, but if another player arrives to a place which has a player with the treasure, the arriving player gets the treasure. When a player with the treasure arrives to the starting place, that player wins the game and the game ends. After the treasure has been found any player with a horse shoe can win by arriving at the starting place before the player with the treasure.

Verbs in Req. Specification (5)

The game boards, places, pieces, labels, connections, etc. each have a graphical representation in the user interface, but that representation is not specified here.

Some potential associations

- The game is played by players
- Player has a playing piece
- Pieces are in starting place -> Piece is in a place
- Pieces are moved on the playing board
- Game has (a set of places) -> map
- Places are connected by a connection
- Connections form paths on the board
- Places are either ordinary places or special places
- Game has a die
- Player may use a flight connection
- Player is at one end of the flight connection
- Special place has a card
- Player throws the dice
- Player has the treasure

Reasons for rejecting associations

- Association may be trivial
- Association may be irrelevant
- Association may be follow from other associations
- Association may turn out to be a momentary action - in this case we may find an operation
- Assocation may relate several classes, and it should be split into binary associations.
- Potential association may actually be an inheritance
- Common sense should (also) be used

Rejecting Associations

1. The game is played by players
2. Player has a playing piece
3. Piece is in a place
4. Pieces are moved on the playing board - activity
5. Game has a map
6. Places are connected by a connection –>
   map contains places and connections
7. Connections form paths on the board – map info
8. Places are either ordinary places or special places
   - inheritance
9. Game has a die
10. Player may use a flight connection - activity
11. Player is at one end of the flight connection (3)
12. Special place has a card
13. Player throws the dice - activity
14. Player has the treasure

Association names

1. The game is played by players ⇒ play
2. Player has a playing piece ⇒ uses
3. Piece is in a place ⇒ located
4. Game has a map ⇒ has
5. Places are connected by a connection ⇒ connect
6. Map contains places and connections ⇒ contains
7. Game has a die ⇒ includes
8. Special place has a card ⇒ Card covers special place ⇒ covers
9. Player has the treasure ⇒ owns

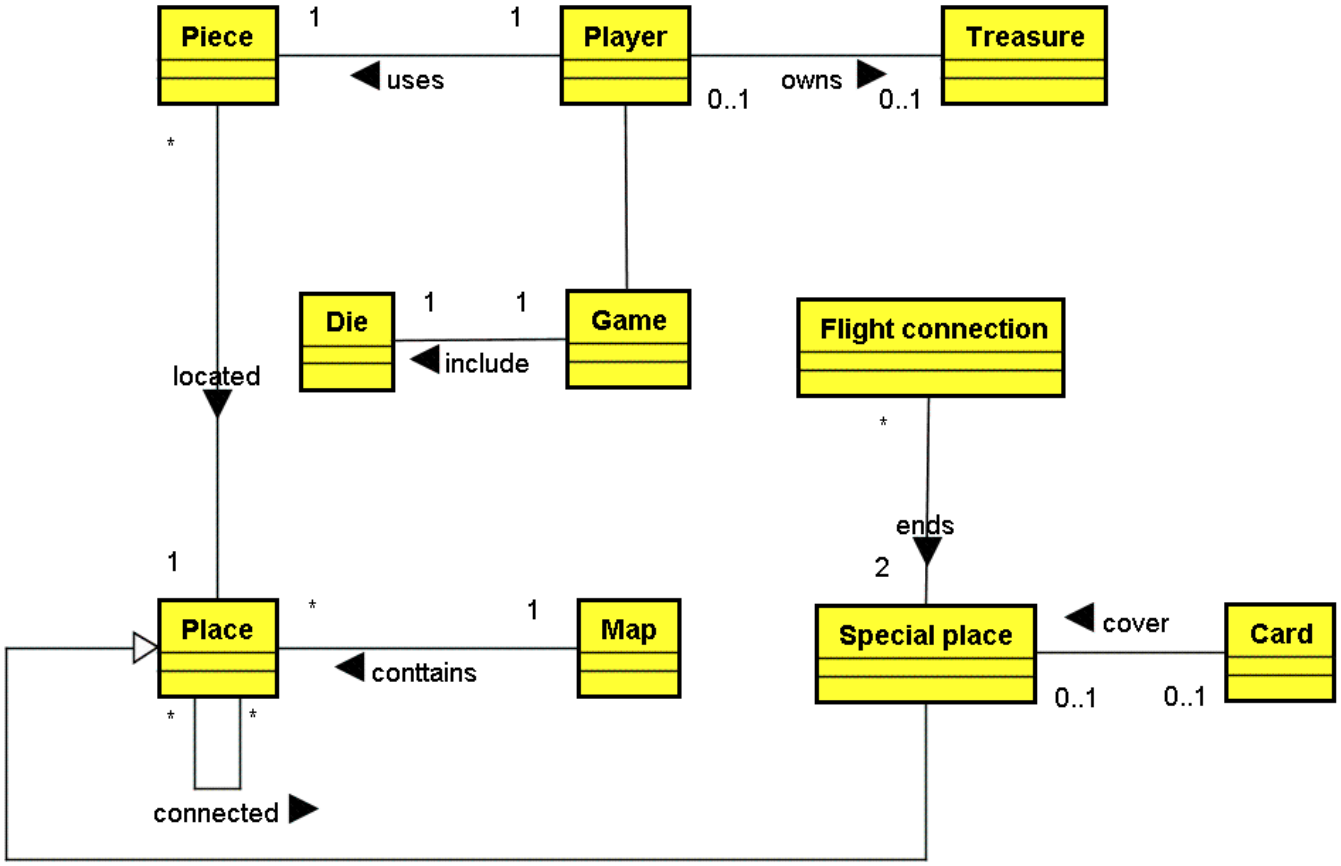
Association names

1. player–play–game
2. player-use-piece $\Rightarrow$ uses
3. piece-located-place $\Rightarrow$ are
4. game-has-map $\Rightarrow$ has
5. connection-connect-places $\Rightarrow$
   - we decide to make connection an association between place and place: place-connected-place
6. map-contain-places
7. game-include-die
8. card-cover-special place
9. player-own-treasure

Association details

- Find good and short names for associations.
- Add extra specifications, such as
  - Navigation direction,
  - Aggregation information,
  - Ordering information,
  - Multiplicity specifications, and
  - Qualifiers (used to identify objects of target end of the association)

Classes and Associations



Class Responsibilities

At this point the roles of the classes could be checked i.e. why the classes are there:

Piece	maintains player's position on the map
Die	gives a random number from {1,2,3,4,5,6} with equal probability (1/6)

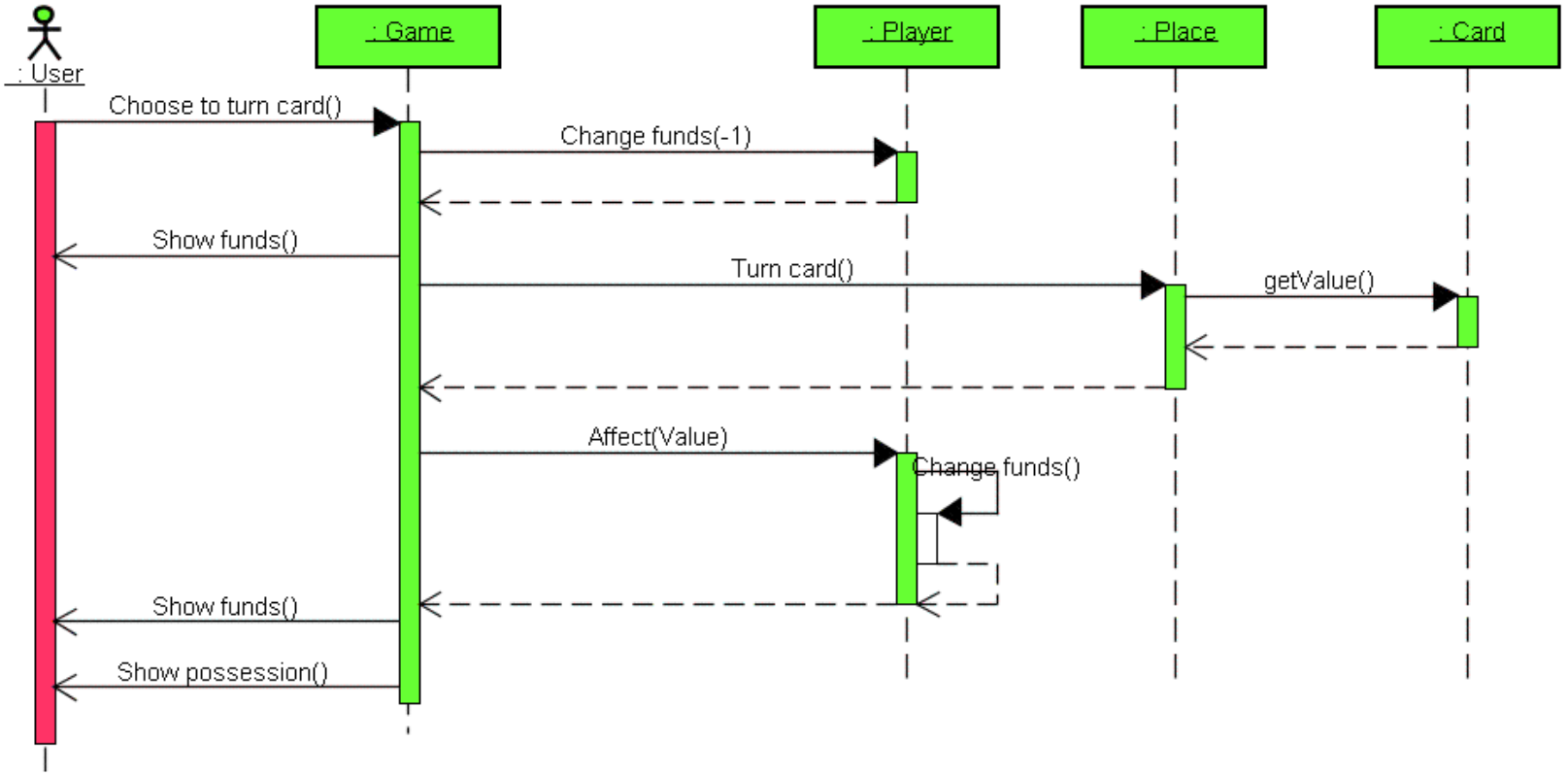
...

...

Deeper Sequence Diagrams

- After having identified the classes more detailed sequence diagrams on the system behaviour can be drawn
- Sequence diagrams use the objects of classes in the class diagram
- Flow of operations from object to object is described

Sequence Diagram: ”Turning Card”



Identifying Operations

Operations come primarily from three sources:

- Sequence diagrams each message arrow represents a call
- Requirement specification: the potential operations are likely to be verbs
- Task lists collected earlier on (this is actually just a double check – all of these should be in the sequence diagrams.)

Some Operations for Class ”Player”

- from the previous sequence diagram we get
  - ChangeFunds(amount:integer):integer and Affect(value:Value):Possessions
- from the requirement specifications we get:
  - getPlace():Place hasMoney(value:Integer):Boolean
  - moveTo(p:Place) hasTreasure():Boolean, takeTreasure(t:Treasure) giveTreasure(t:Treasure) isWinner():Boolean, canFly():Boolean

Identify Attributes for Classes

- Attributes come from various sources:
  - Requirement specification: it can be found that some attribute is vitally important for a class, like the amount of money for a player.
  - Sequence diagrams may imply a need for an attribute, like the name of a player
- First the attributes that are clearly important for the model are recorded
- In the beginning it is good to avoid such implementation-specific attributes, which may change when a more detailed model is introduced

Inheritance Structures

- Is some class a special case of another class?
- Do we have some kind of a division into different subtypes?
- A compound noun (consisting of more than one words) may indicate inheritance.
- Do we have classes, which seem to share properties?
- Would it be useful to use some generally known wider concept?

Final Check-up

- Is everything necessary?
- Do we need more classes?
- Do we need to split or combine classes?
- Is everything consistent?
- Have we taken into accountall the knock-on effects that our changes have created?
- Does the model seem sufficient? (Does it include all relevant concepts and associations?)

Final Check-Up

- Does the model seem understandable?
- Is the model easy to read? Are the names chosen correctly?
- Have we avoided redundancy? (One thing only done once.)
- Does the model really describe the target system?

Next steps

- The model produced in the above phases serves as a starting point of detailed design, which produces i.a. a class model that can be implemented with the chosen programming language and technology
- Every application also needs a user interface, which has been omitted from the previous example