Sprint 7: Property

Developer A

I Lav en feature branch: `sprint7_buy_[par]`

Koden I bygger nu, vil gælde for alle subklasserne til `Property` `(Plot, ShippingLine eller Brewery)`

I `Property` klassens `onAccept` metode, skal koden tjekke hvad der er blevet accepteret:

- Tjek om `option` er sat til "`buy`"

Hvis `option` er sat til "`buy`" :

- `buyProperty` metoden kaldes på `player` parameteret.
- `owner` sættes til `p` (findes i parameterlisten)
- Kald `checkForMonopoly` metode, (ikke bygget, så udkommenter)
- Returner besked om at spilleren har købt ejendommen.

`onReject` metoden, vil blive kaldt fra `Game`, hvis spilleren har svaret nej til at købe en ejendom

Returner en passende besked om hvad der er blevet afvist,
Fx. "`Egon afviste at købe Rådhuspladsen`"

Muligt at der senere også skal igangsættes en auktionsrunde, men den venter vi lige med…

Developer B

Skift til `Plot` klassen

- Tilføj en attribut `buildings` (inkl. getter)
- I `onLand` metoden, del koden lidt op:

```
String msg = super.onLand(p);
…
return msg;
```

- Efter kald til `super` og før vi returnerer `msg`, skal vi fange om der er monopol på feltet. Hvis ja: `option` sættes til "`build`".

- I `onAccept` metoden på `Plot` klassen, skal vi fange om `option` er sat til "`build`". Hvis ja:
- `buildings` attribut tælles op med 1.
- Spillerens `build` metode kaldes, med husprisen* som argument.
- Tilføj passende besked til `msg`.

** Nogen skal analysere skødekort og finde ud af hvordan husprisen kan beregnes. Brug evt. en dummy værdi.*

Sprint 7: Property

Developer A

I Lav en feature branch: `sprint7_build_[par]`

Tilføj metoden `checkForMonopoly(p)`.

- initialiser en variabel, `seriesSize` med værdien 3. Det er størrelsen på de fleste serier.
- Skriv kode der fanger undtagelserne og overskriver `seriesSize`:

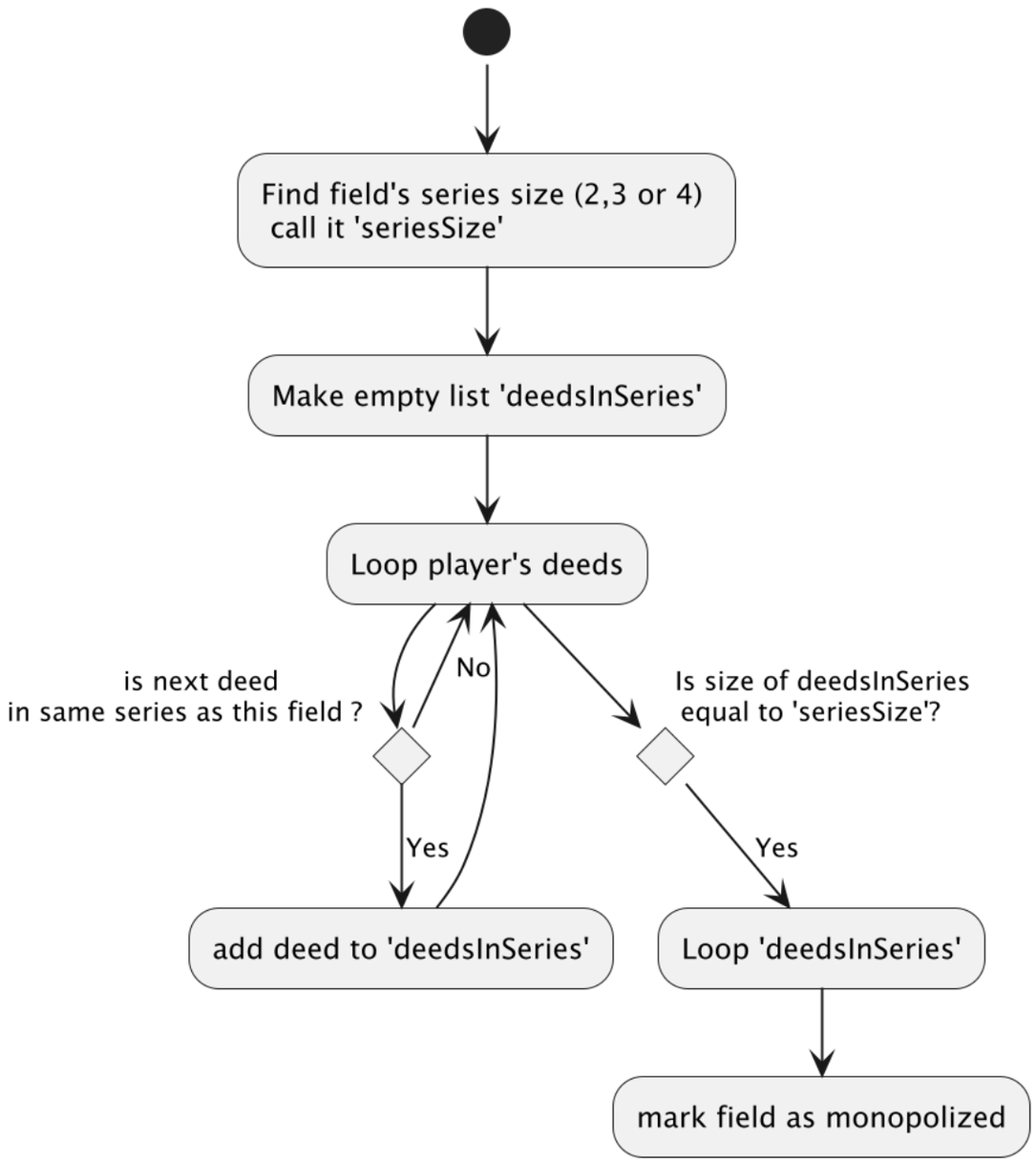
serie 0, 1 og 9: `seriesSize = 2`
serie 3 : `seriesSize = 4`

- initialisér en ArrayList, `deedsInSeries`.
- Lav et loop hvor du gennemløber listen af spillerens skøder.
- For hver iteration, check om feltet har samme serie ID, som det felt metoden kører på (`this`). Hvis det er tilfældet, læg feltet i `deedsInSeries`

Efter loopet

- Tjek om størrelsen på `deedsInSeries` er det samme som `seriesSize`
- Gennemløb felterne i `deedsInSeries`, for at sætte alle felternes `isMonopolized` til `true`.

Developer B



Sprint 7: Dice

Developer A

Lav en feature branch: `sprint7_Dice_[par]`

Tilføj en `isDouble` attribut til `Dice` klassen, giv den værdien `false`.

Sæt den til `true` hvis der bliver slået dobbelt slag nede i `rollDiceSum` metoden, og til `false` hvis det modsatte er tilfældet

I Game,

- Tilføj en `doubleDiceCounter` til klassen
- I `throwAndMove`, efter feltet er identificeret, indsæt et tjek på om `dice.isDouble` er `true`.

Hvis `true`:

- tæl `doubleDiceCounter` op med én,
- Indsæt en nested betingelse som tjekker at `doubleDiceCounter` er mindre end 3.

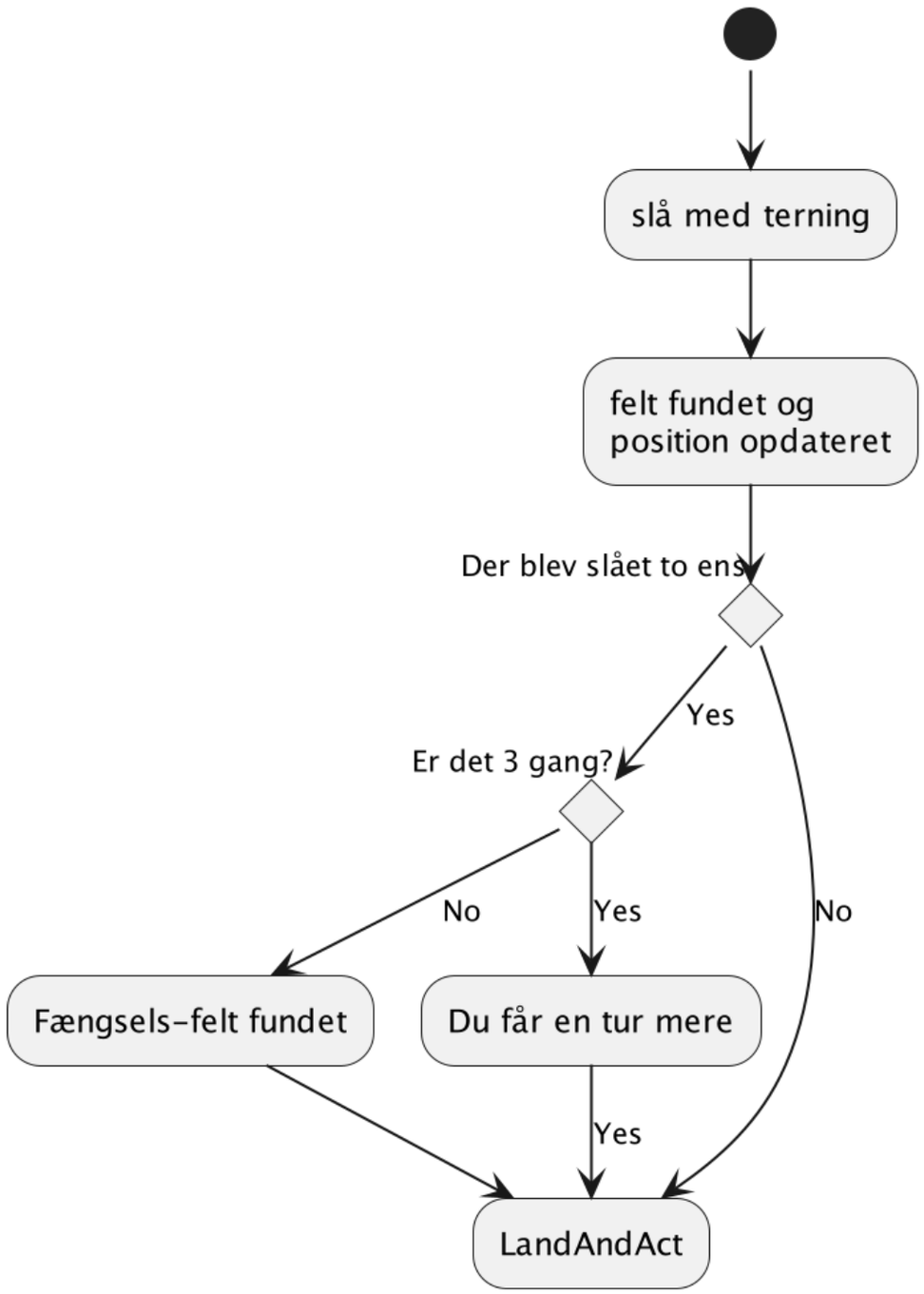
  Hvis `true`:
  - Vis en besked til brugeren om at spilleren får en ekstra tur.
  - Tæl `count` (den der tæller hvis tur det er), ned med én.

  Hvis `false`:
  - Vis en besked til brugeren om at han har slået for mange dobbeltslag.
  - Sæt `doubleDiceCounter` til 0.
  - Få fat I `Prison` feltet og overskriv `f`:

```
f = board.getField(31);
```

Du kan teste ved at pille ved rollDiceSum - erstat de to kald til random metoden, med ens værdier.

Developer B



Push feature branch til git

Sprint 7: Chance

Developer A

I Lav en feature branch: `sprint7_Chance_[par]`

I `onAccept`, læs pseudo koden og implementér:

```
public String onAccept(Player p){

/*
Switch(card.event)

case "pay": kald player's pay metode med cost som argument.
case "receive": kald player's receive metode med income som argument.
case "prison": kald player's moveToPosition metode med 11 som
argumentet
case "wildcard": kald player's setWildcard metode med true som
argument*/

Osv
*/
 return card.getMessage();
}
```

Obs! Der skal være et match mellem cases og events i carddata. Tjek data/cardsdata.csv: Koordiner med data folket hvis I er i tvivl.

Hvis I opdager et kort der giver spilleren en valgmulighed, skal der returneres en passende besked til bruger, og `Field` klassens `option` attribut skal sættes.

Developer B

Kun relevant hvis der findes kort som giver spilleren en valgmulighed.

I `onAccept` metoden kan du skrive `case` blokke der håndterer eventuelle valgmuligheder I fandt.

Push feature branch til git

Sprint 7: Player

Developer A

Implementer nogle flere player metoder:

`moveToPosition(int value)`:
- Sæt `position` til value.

Tilføj attributten `boolean hasWildcard`, incl. en getter og en sætter:
- `boolean getWildcard()`: returnerer `hasWildcard` (attributten)
- `void setWildcard(boolean value)`: sætter `hasWildcard` til `value`

Developer B

`imprison();`
- Tilføj attribut, `inPrison`.
- Sæt `inPrison` til `true`.
- Kald `moveToPosition` med værdien 31 som argument

Når vi skal betale for at lande på et bryggeri med monopol, har vi brug for at vide hvad der blev slået. Det ved vi i `updatePosition` metoden.

- sørg for at denne værdi bliver gemt i en klasse attribut, så man kan tilgå den fra `Brewery` klassen - i tilfælde af at spilleren lander der.

Sprint 7: Prison

Developer A

Lav en feature branch: `sprint7_prison_[par]`

Opret klassen `Prison` klassens `onLand` metode:

Gem returværdien af et kald til `super.onLand` i en `String msg`.
Tilføj flg. tekst til `msg`:

```
"\n Du er arresteret, og skal i fængsel”
```

- Tjek om spilleren har et wildcard. Ved at kalde spillerens `getWildcard()` metode.
- Hvis kaldet returnerer `true`, tilføje flg. tekst til `msg`:
  `"\n Vil du bruge dit wildcard?(Y/N)”`
- Hvis kaldet returnerer false, skal du kalde spillerens `moveToPosition` med værdien 31 som argument.
- Tilføj tekst til `msg` der forklarer hvordan han kommer ud af fængslet igen *.
- Returner `msg`.

Developer B

`onAccept(Player p)`

- Kald spillerens `setWildcard` metode med `false` som argument.
- Tilføj tekst til `msg`: `”\n Du har kontakter hos Politiet og løslades.”`

`onReject(Player p)`

- Returner en besked om at spilleren ryger i fængsel, kald spillerens `moveToPosition` med værdien 31 som argument.
- Tilføj tekst til `msg` der forklarer hvordan han kommer ud af fængslet igen *.
- Returner `msg`.

Reglerne for hvordan man kommer ud af fængslet kræver måske lidt research.
Find en domæneekspert og spørg, om dette er passende besked:

```
*  "\n Næste gang det bliver din tur, kan du
 vælge om du vil betale dig ud
 eller prøve at slå et dobbeltslag.
 Når du har siddet over i tre omgange er du løsladt”
```