Sprint6: Board

Developer A

I Lav en feature branch: `sprint6_board_[gruppe]`

I `createFields` metoden,

- der hvor du henter data ud af `values` arrayet, sørg for også at gemme feltets type i en variabel, `fieldtype`

-Ret linjen:
`Field f = new Field(…)` til `Field f = null;`

-På linjen lige efter, lav en `switch-case` blok, der evaluerer variablen `fieldtype`.

-Kig på `fielddata.csv` og find ud af hvad data folket har valgt at kalde de forskellige typer.

```
switch (fieldtype) {
    case “Chance":
                break;
    case "Start":
                break;

Osv…

    default: System.out.println(“No such type available”);
}
```

Developer B

I den enkelte `case` blok, instansier den specifikke felt type:

Eksempel:
```
case “Plot": f = new Plot(ID, label, income, cost);
                break;
```

Push feature branch til git

Sprint 6: Player

Developer A

Lav en feature branch: `sprint6_player_[par]`

Se `Player` klassen i seneste klassediagram

I de næste trin skal I tilføje flere metoder til `Player` klassen

`pay(int amount)`
Kald `Account` instansens `withdraw` metode med `amount` som argument.

`buyProperty(Field f):`

- Træk feltets pris fra spillerens konto, ved at kalde `pay` metoden med feltets pris som argument.
- Tilføj og initialiser `deeds` attributten.
- Læg `f` i `deeds`.

Developer B

`pay(int amount, Player recipient)`

Metoden bruges når en spiller skal betale et beløb til en anden spiller. Metoden skal indeholde:

- Et kald til `pay` metoden med `amount` som argument.
- Et kald til modtagerens `receive` metode med `amount` som argument: `recipient.receive(amount)`

`getTotalWorth()`

- Læg prisen på alle felter i `deed` listen sammen og til sidst lægger `balance` på spillerens konto til resultatet
- Returner der samlede resultat.

Push feature branch til git

Sprint6: Property

Developer A

Lav en feature branch: `sprint6_Property_[gruppe]`

I skal skrive kode der gør, at en spiller modtager et tilbud om at købe når han lander på en ejendom.
I den sidste task, skal I starte på kode der håndterer at spilleren skal have tilbud om at bygge, hvis man lander på en grund med monopol.

Fortsæt `if-else` blokken.

`true`: **ingen** ejer ejendommen,
- sæt `option` til "buy"
- tilføj flg. til `msg`:

```
msg+= "Vil du købe?"(Y/N):"
```

`false`: Tjek om det er en anden en ejere selv, der er ejeren
Hvis **en anden** ejer ejendommen
- Læg mere information til `msg`:

```
msg+= "Du skal betale "+income+"
```

- Kald dernæst spillerens `pay` metode med feltet (`this`) som argument.

(Obs: Metoden findes ikke endnu, udkommenter, for at compile)

Developer B

- I `Property` klassens `onLand` foretag flg. ændring:

```
return super.onLand();
```

Bliver til…

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

- Imellem erklæring og returnering af `msg`, lav et `if-else` statement, der tjekker om `owner` er `null`

- I `Plot` klassens `onLand` metode, foretag flg. ændring:

```
return super.onLand();
```

Bliver til…

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

- Imellem erklæring og returnering af `msg`, lav et `if-else` statement, der tjekker om `owner` er `p`
OG om der er monopol på feltet (kald `checkForMonopoly()`)

(Obs: Metoden findes ikke endnu, udkommenter, for at compile)

`true`: **spilleren** ejer ejendommen OG har monopol,
- sæt `option` til "build" og tilføj flg. til `msg`:

```
msg+= "Vil du bygge?"(Y/N):"
```

Sprint 7: Tax

Developer A

Lav en feature branch: `sprint6_tax_[par]`

- Opret alle subklasserne til Field. Se seneste klassediagram Bemærk at Felter som giver spilleren en valgmulighed.
- Tilføj et interface IOption med metoderne `onLand` og `onReject`
- `Property, Prison` og `Tax`, skal implementere `IOption` og derfor have `onLand` , `onAccept` og `onReject` metoder.

Undlad at implementere koden i metoderne. Sørg bare for at de kan compile. Ellers kan klasserne være tomme.

I Tax klassens `onAccept` metode

- Kald spillerens `pay`-metode med feltets `cost` attribut som argument.
- Returner en String med en besked der bekræfter hvad der lige er sket.

Developer B

I `Tax` klassens `onLand` metode:

- Gem returværdien af et kald til `super.onLand` i en `String s`.

- Tilføj flg tekst til `s`:

```
"\n Du skal betale et beløb der svarer til 10 % af dine aktiver. Vil du hellere betale et fast beløb på "+this.cost+"kr? Y/N \n”;
```

- Returner `s`.

I Tax klassens `onReject` metode

- Beregn 10 % af spillerens værdi. Spillerens værdi hentes ved kaldet: `p. getTotalWorth()`

- Kald spillerens `pay`-metode med resultatet.

- Returner en String med en besked der bekræfter hvad der lige er sket.