# Mahjong Analyzer

Progress 2

Muhammad Jilan Wicaksono

November 3, 2025

Universitas Indonesia

# Target dan Pencapaian

1. **Menyusun tipe data:** 100%

   untuk Suit, Tile, Hand, Meld, dan Agari.

2. **Membuat fungsi utilitas:** 100%

   untuk mencari Pair dan Triplet, serta membuat parser sederhana.

3. **Membuat fungsi validasi kemenangan:** 75%

   baik standar maupun non-standar.

   Target bertipe `agariCheck :: Hand -> Maybe Partition`

   tetapi masih bertipe `agariCheck :: Hand -> Bool`

# Link Commit

- Untuk agariCheck: Membuat fungsi validasi tangan sederhana.
- Untuk fungsi utilitas: Membuat ADT untuk entitas permainan serta menambahkan parser sederhana.

# Lesson Learned

1. ADT untuk `Tile, Pair, Meld` agar type-safe.
2. Higher-Order Function untuk fungsi `removeTiles` menggunakan `removeOneTile`.
3. Recursion dalam pemanggilan `agariCheck`.
4. `Either Monad` pada parser untuk error handling.

# Lesson Learned: ADT i

```haskell
data Suit = Manzu | Pinzu | Souzu | Honor
    deriving (Show, Eq, Ord, Enum, Bounded)

data Tile = Tile Suit Int
    deriving (Show, Eq)

instance Ord Tile where
    compare :: Tile -> Tile -> Ordering
    compare (Tile s1 n1) (Tile s2 n2)
        | s1 == s2    = compare n1 n2
        | otherwise   = compare s1 s2
```

# Lesson Learned: ADT ii

```haskell
data Meld
    = Sequence Tile          -- Sequence M4 : M4, M5, M6
    | Triplet Tile           -- Triplet Z3 : Z3, Z3, Z3
    deriving (Show)

data PMeld
    = MissingMiddle Tile     -- MissingMiddle P4 : ada P4 dan P6
    | MissingOut Tile        -- MissingOut S2 : ada S2 dan S3
    deriving (Show)

newtype Pair = Pair Tile  -- Pair P3 : P3, P3
```

# Lesson Learned: Higher-Order Function i

```
1  removeOneTile :: Tile -> HandCount -> HandCount
2  removeOneTile =
3      let updateFunc c = if c > 1 then Just (c - 1) else
           Nothing
4          in Map.update updateFunc
5
6  removePair :: Pair -> HandCount -> HandCount
7  removePair (Pair tile) hc =
8      let tilesToRemove = replicate 2 tile
9          in foldl' (flip removeOneTile) hc
               tilesToRemove
```

# Lesson Learned: Higher-Order Function ii

```
1  removeSequence :: Meld -> HandCount -> HandCount
2  removeSequence (Sequence (Tile suit num)) handCount =
3      let tilesToAdjust = [Tile suit num, Tile suit (num
           + 1), Tile suit (num + 2)]
4
5      in foldl' (flip removeOneTile) handCount
           tilesToAdjust
6
7  removeTriplet :: Meld -> HandCount -> HandCount
8  removeTriplet (Triplet tile) handCount =
9      let tilesToAdjust = replicate 3 tile
10
11     in foldl' (flip removeOneTile) handCount
           tilesToAdjust
```

# Lesson Learned: Recursion i

```
checkMeld :: HandCount -> Int -> Bool
checkMeld hc meldToFind
  | meldToFind == 0 = Map.null hc
  | otherwise =
    case Map.minViewWithKey hc of
      Nothing -> False

      Just ((firstTile, count), _) ->

        let tryTriplet =
          (count >= 3) &&
          let m = Triplet firstTile
            hc' = removeTriplet m hc
              in checkMeld hc' (meldToFind - 1)

```

# Lesson Learned: Recursion ii

```
16            trySequence =
17              isSequence firstTile hc &&
18              let m = Sequence firstTile
19                hc' = removeSequence m hc
20              in checkMeld hc' (meldToFind - 1)
21
22          in tryTriplet || trySequence
```

# Lesson Learned: Monad i

```haskell
parseHand :: String -> Either String Hand
parseHand "" = Right []  -- Base case: tangan kosong
parseHand str =
  -- Pisahkan grup angka di depan
  -- span isDigit "123m456p" -> ("123", "m456p")
  let (nums, rest) = span isDigit str
  in
    if null nums then
      Left "Input tidak valid: Diharapkan ada angka."
    else if null rest then
      Left "Input tidak valid: Diharapkan ada suit (m,
          p,s,z) setelah angka."
    else
      let suitChar = head rest
        remainingString = tail rest
```

# Lesson Learned: Monad ii

```
15          in
16            case charToSuit suitChar of
17              Left err   -> Left err
18              Right suit ->
19                -- Buat tiles untuk bagian ini
20                let currentTiles = stringToTiles nums suit
21                -- Panggil rekursi untuk sisa string
22                in (currentTiles ++) <$> parseHand
                      remainingString
```