



Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB V

Implementasi dan Evaluasi

Setelah mempelajari kelas yang merupakan referensi dari internet, kelas – kelas tersebut diimplementasikan dan diadaptasikan dengan lingkungan Java i-mode. Berikut adalah beberapa kode hasil implementasi berdasarkan referensi:

Kelas *Game*

Instansi dari kelas ini mengatur jalannya permainan Mastermind. Kelas ini memiliki 1 atribut yaitu *pegs_pattern* yang menyimpan pola permainan yang harus ditebak. Kelas ini juga memiliki 2 fungsi yaitu *Generate()* dan *Compare()*. Fungsi *Generate()* bertugas untuk membuat pola secara acak dengan cara menggunakan fungsi *Random()* yang terdapat dalam bahasa pemrograman Java dan menempatkan hasil yang sudah diacak pada *pegs_pattern*.

```
pegs_pattern = new in
for (int i = 0; i < 4
do {
    s = "" + r.ne
} while (v.contai
v.addElement(s);
pegs_pattern[i] =
```

Gambar 5. 1. Potongan fungsi *Generate()*

Fungsi *Compare()* bertugas untuk membandingkan pola yang ditebak dengan pola yang dibuat dan juga memberikan penilaian berupa banyaknya pasak hitam atau putih setiap kali pemain menebak pola yang dibuat.

```

for (i = 0; i < 4; i++)
    checkBlack[i] = false;

for (i = 0; i < 4; i++)
    if (pegs_pattern[i] == guess_peg[i])
        checkBlack[i] = true;
        blackNum++;
    }

if (blackNum < 4) {
    boolean[] checkWhite = new boolean[4]
    for (i = 0; i < 4; i++)
        checkWhite[i] = checkBlack[i];
    for (i = 0; i < 4; i++)
        if (!checkBlack[i])

```

Gambar 5. 2. Potongan fungsi *Compare()*

Kelas *GuestResult*

Instansi dari kelas ini menyimpan hasil tebakan yang dibuat oleh kelas *Game*. Kelas ini memiliki 3 atribut yaitu *black_count*, *white_count*, dan *succeed*. *black_count*, *white_count* menunjukkan banyak pasak hitam atau putih berdasarkan hasil penilaian kelas *Game* dan *succeed* mengindikasikan berhasil atau tidaknya pemain menebak pola setiap kali pemain menebak pola yang sudah dibuat.

```

class GuestResult {

    private int black_count, white_count;

    public int getBlack_count() {
        return black_count;
    }

    public int getWhite_count() {
        return white_count;
    }
    boolean succeed;
}

```

Gambar 5. 3. Kelas *GuessResult*

Kelas *ObjectScore*

Instansi dari kelas ini menyimpan data untuk penghitungan nilai akhir. Kelas ini memiliki 2 atribut yaitu *guess* dan *time* yang masing – masing digunakan untuk menyimpan banyak tebakan dan waktu yang diperlukan saat memecahkan suatu kode.

```
class ObjectScore {  
    private int guess, time;  
  
    public ObjectScore(int gue  
        this.guess = guess;  
        this.time = time;  
    }  
}
```

Gambar 5. 4. Kelas *ObjectScore*

Kelas *ScorePanel*

Kelas ini bertugas untuk menyimpan dan menampilkan nilai dari 5 pemain yang mendapatkan nilai paling tinggi. Ada 2 fungsi utama dari kelas ini yaitu *score_read()* dan *score_write()* yang dimana *score_read()* berfungsi untuk membaca nilai dari *Scratchpad* dan *score_write()* berfungsi untuk menulis nilai ke *Scratchpad*. Fungsi yang lain adalah *sort()* yang bertugas untuk mengurutkan data nilai yang akan ditampilkan.

```
private ObjectScore[] score_read(int count  
    ObjectScore[] tmp = new ObjectScore[co  
    DataInputStream dis = null;  
    try {  
        dis = Connector.openDataInputStrea
```

Gambar 5. 5. Potongan fungsi *score_read()*

```

public void score_write(ObjectSco
    DataOutputStream dos = null;
    ObjectScore[] osRead = score_
    int length = 0;
    for (int i = 0; i < osRead.le
        if (osRead[i].getGuess()
            length++;
    if (length < 5)

```

Gambar 5. 6. Potongan fungsi *score_write()*

```

for (i = 0; i < n; i++)
    os[i].setScore((9
for (i = 0; i < n; i++)
    for (j = 0; j < n .
        if (os[j].getSc

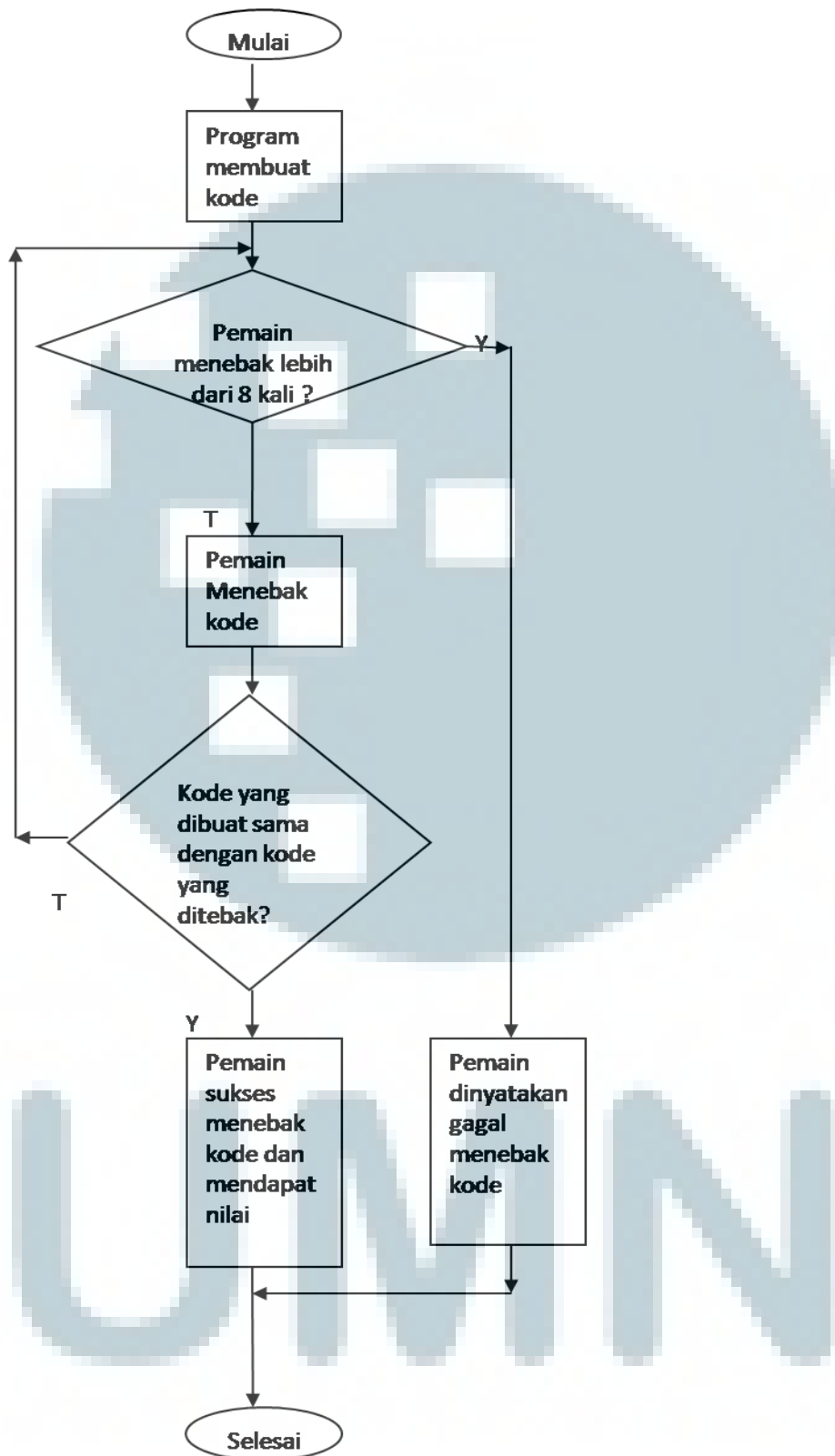
```

Gambar 5. 7. Potongan fungsi *sort()*

Masih ada beberapa kelas di permainan ini dan kelas – kelas tersebut diimplementasikan oleh rekan kerja yaitu Ivan Prakasa.

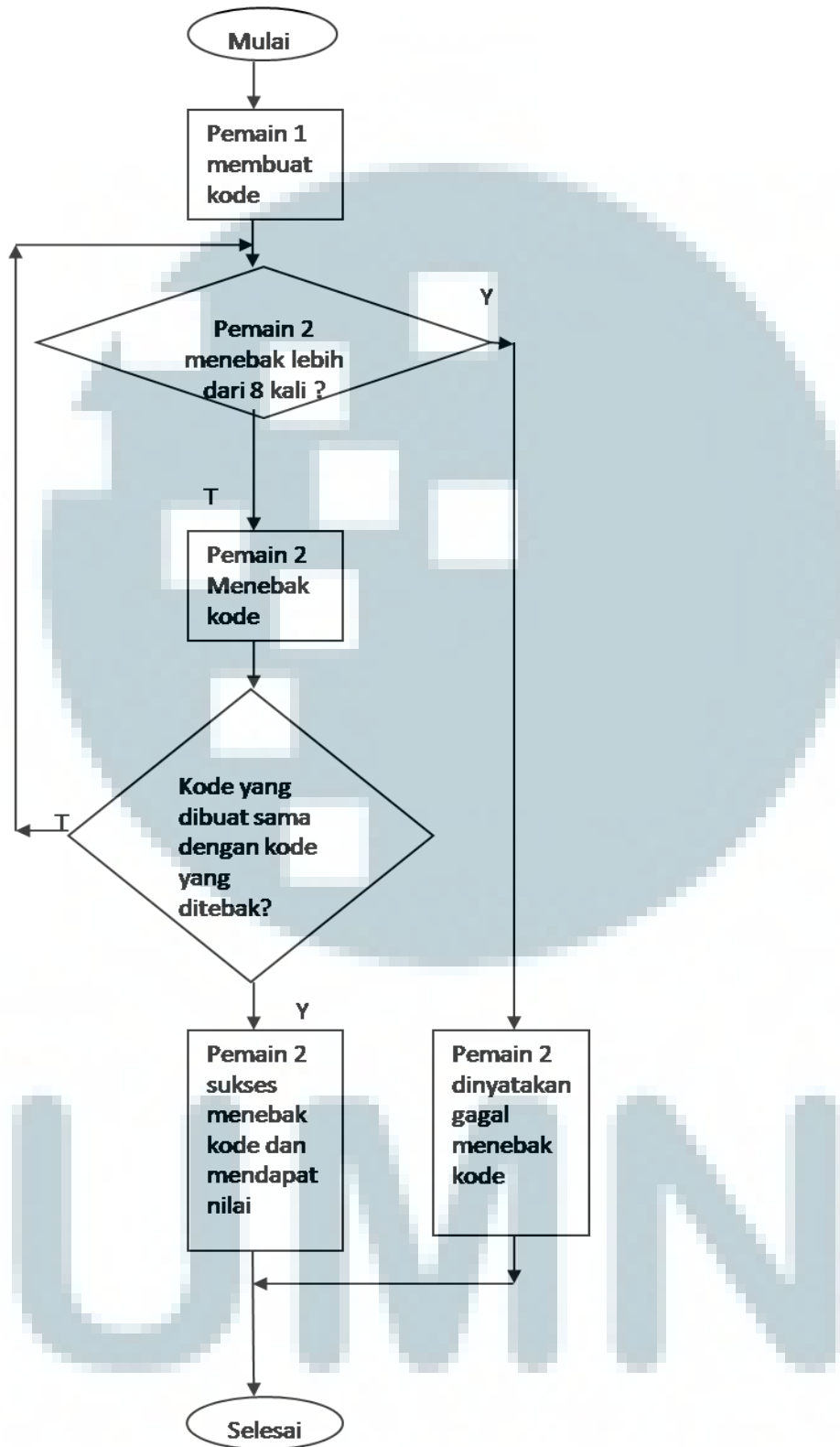
UMMN

Flowchart permainan untuk 1 orang pemain :



Gambar 5. 8. Flowchart permainan untuk 1 orang pemain

Flowchart permainan 2 orang pemain :



Gambar 5. 9. Flowchart permainan untuk 2 orang pemain

Pseudo code selama permainan berlangsung :

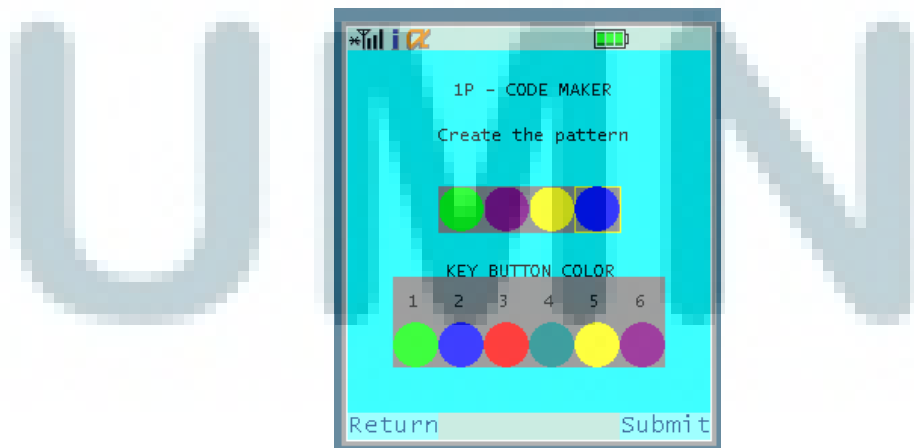
Mulai

1. Kode dibuat oleh program oleh pemain pertama;
2. Pemain menebak kode yang sudah dibuat;
3. Jika kode yang dibuat cocok dengan kode yang ditebak maka pemain berhasil memecahkan kode tersebut;
4. Jika kode yang dibuat belum cocok maka program akan memberikan penilaian berupa pasak hitam atau putih sebagai pembantu untuk menebak kode yang tepat;
5. Jika pada kesempatan ke delapan pemain belum bisa menebak kode dengan tepat, maka pemain dinyatakan gagal;

Selesai

Sebagai contoh:

Kode yang dibuat adalah maka tampilan pada layar akan seperti gambar berikut :

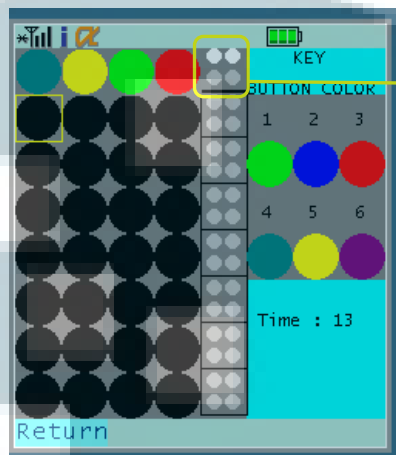


Gambar 5. 10. Kode yang dibuat pemain pertama

Pemain menebak

④ ⑤ ① ③

Maka program akan memberikan penilaian berupa 2 pasak putih yang berarti terdapat 2 pola yang ditebak dengan tepat namun posisinya masih salah. maka tampilan pada layar akan seperti gambar berikut :



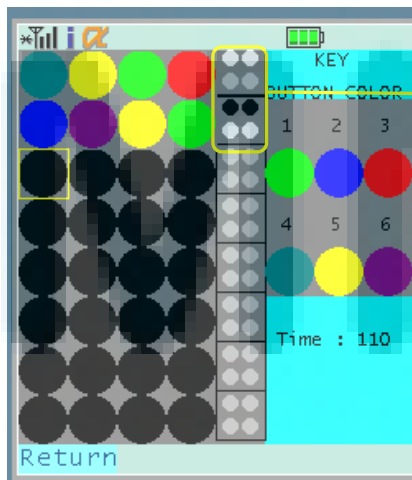
Pasak penilaian berupa putih / hitam

Gambar 5. 11. Hasil tebakan pertama

Pemain menebak

② ⑥ ⑤ ①

Maka program akan memberikan penilaian berupa 2 pasak hitam dan 2 pasak putih yang berarti semua pola ditebak dengan tepat namun terdapat 2 pola yang posisinya masih salah. Berikut adalah hasil tampilannya :



Pasak penilaian berupa putih / hitam

Gambar 5. 12. Hasil tebakan kedua

Pemain menebak

① ⑥ ⑤ ②

Maka program akan memberikan penilaian berupa 4 pasak hitam dan pemain dinyatakan berhasil memecahkan kode dan mendapatkan skor. Hasilnya akan terlihat seperti gambar berikut ini :



Gambar 5. 13. Hasil tebakan ketiga

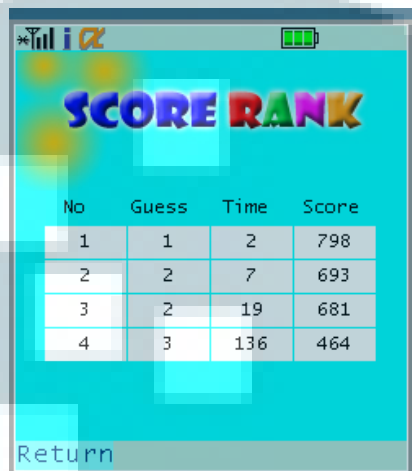
Penghitungan Nilai Akhir sebagai berikut

Nilai Akhir = $(9 - \text{Banyaknya tebakan}) * 100 - \text{lamanya menebak dalam detik}$

Jika Nilai akhir < 0 maka Nilai akhir menjadi 0 yang berarti nilai terendah jika pemain berhasil memecahkan kode adalah 0.

UMMN

Setiap saat pemain berhasil memecahkan kode yang telah dibuat, maka skor dari pemain akan dicatat di tabel skor. Tabel skor merupakan hasil implementasi kelas *ScorePanel*. Tabel skor hanya menampilkan 5 pemain yang mendapatkan skor tertinggi. Tampilan tabel skor seperti gambar berikut ini :



No	Guess	Time	Score
1	1	2	798
2	2	7	693
3	2	19	681
4	3	136	464

Gambar 5. 14. Tampilan kelas *ScorePanel*

Evaluasi

Setelah kedua proyek (Proyek desain tampilan dan Proyek algoritma permainan Mastermind) digabung, program yang dibuat sudah bisa dipakai dan sudah diuji semua fungsionalitasnya. Pengujian juga sudah dilakukan dalam penghitungan nilai akhir untuk pemain yang berhasil menebak kode dengan tepat.