

**UJIAN AKHIR SEMESTER
PEMROGRAMAN BEORIENTASI OBJEK**



Disusun Oleh:

Kelompok 1

- | | |
|-----------------------------|-------------|
| 1. Esra Silvia Sihite | (G1F022035) |
| 2. Muhammad Rifky Ramadani | (G1F022039) |
| 3. Muhammad Salman Alfarizi | (G1F022047) |

Asisten Dosen:

- | | |
|-------------------------|-------------|
| 1. Randi Julian Saputra | (G1A019066) |
|-------------------------|-------------|

Dosen Pengampu:

1. Kurnia Anggriani, S.T., M.Eng
2. Arie Vatesia, S.T., M.T.I., Ph.D.

**PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNIK UNIVERSITAS
BENGKULU
2023/2024**

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan rahmat dan karunianya, sehingga kami dapat menyelesaikan Laporan Pelaksanaan Ujian Akhir Semester ini. Ujian Akhir Semester ini merupakan salah satu kegiatan akhir yang wajib dilakukan mahasiswa untuk menentukan nilai akhir di matakuliah yang ada di Prodi Sistem Informasi Fakultas Teknik Universitas Bengkulu. Dengan selesainya laporan Ujian Akhir Semester ini tidak terlepas dari bantuan banyak pihak yang telah memberikan masukan-masukan kepada kami. Untuk itu kami mengucapkan banyak terimakasih. Kami menyadari bahwa penyusunan laporan Ujian Akhir Semester ini masih jauh dari sempurna. Mengingat keterbatasan waktu dan kemampuan kami. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang bersifat membangun, demi kesempurnaan laporan Ujian Akhir Semester ini. Akan tetapi, kami berharap semoga laporan ini dapat bermanfaat, khususnya bagi kami.

Bengkulu, Desember 2023

Kelompok 1

DAFTAR ISI

COVER	i
KATA PENGANTAR	ii
DAFTAR ISI	iii
BAB I	1
PENDAHULUAN.....	1
A. Latar Belakang	1
B. Rumusan Masalah	2
C. Tujuan Dan Manfaat.....	2
BAB II.....	3
LANDASAN TEORI.....	4
1.1 PBO (Pemrograman Berorientasi Objek).....	5
1.2 Game Hangman... ..	5
BAB III.....	6
PEMBAHASAN.....	6
A. Source Code.....	6
B. Analisis Program	7
C. Penjelasan Output.....	13
BAB IV.....	16
KESIMPULAN DAN SARAN	16
A. KESIMPULAN	16
B. SARAN	16
C. DAFTAR PUSTAKA.....	17

BAB I

PENDAHULUAN

A. Latar Belakang

Game Hangman adalah permainan tebak kata yang seru dan populer. Dalam permainan ini, seorang pemain harus menebak kata yang dipilih secara acak oleh pemain lain atau oleh komputer. Kata tersebut biasanya terdiri dari huruf-huruf yang disembunyikan, dan pemain harus menebak huruf-huruf tersebut satu per satu. Papan permainan Hangman biasanya terdiri dari garis-garis yang mewakili huruf-huruf dalam kata yang harus ditebak. Pemain harus menebak huruf-huruf yang mungkin ada dalam kata tersebut. Jika tebakan pemain benar, huruf tersebut akan diungkapkan dalam kata yang disembunyikan, dan jika tebakan salah, pemain akan mendapatkan bagian tubuh gantungan (hangman) secara bertahap. Tujuan pemain adalah menebak kata sebelum hangman lengkap tergambar. Permainan Hangman biasanya dimainkan dengan kata-kata dalam bahasa Inggris, tetapi dapat diadaptasi untuk menggunakan kata-kata dalam bahasa lain. Ini adalah permainan yang sering dimainkan untuk mengasah keterampilan kosakata, menguji imajinasi, dan melatih strategi menebak kata.

Sejarah game Hangman (gantungan) tidak sepenuhnya jelas dan ada beberapa spekulasi tentang asal-usulnya. Permainan ini dianggap sebagai versi modern dari permainan yang lebih tua yang melibatkan menggantung boneka. Salah satu teori menyatakan bahwa game Hangman berasal dari zaman Romawi Kuno. Dalam versi itu, seorang prajurit yang mengkhianati bangsa Romawi digantung secara bertahap setiap kali tebakannya salah. Namun, tidak ada bukti yang meyakinkan untuk mendukung teori ini. Game Hangman dalam bentuknya yang lebih terkenal kemungkinan berasal dari abad ke-17 atau ke-18 di Eropa. Permainan ini dikenal dengan beberapa nama seperti "Gallows", "Hanged Man", atau "Hanging". Pada awalnya, permainan ini dimainkan dengan mencari tahu kata-kata yang berkaitan dengan hukuman gantung. Hangman adalah permainan tebak kata sederhana yang melibatkan pemain mencoba menebak kata atau frasa dengan menebak huruf-huruf yang termasuk dalam kata tersebut. Permainan ini biasanya dimainkan oleh dua orang atau lebih. Permainan Hangman sering digunakan sebagai cara yang menyenangkan untuk meningkatkan kosakata dan pemahaman terhadap kata-kata.

Pada abad ke-19, permainan Hangman mulai muncul dalam bentuk yang lebih mirip dengan yang kita kenal sekarang. Kata-kata misteri digunakan dan pemain harus menebak huruf-hurufnya untuk mengungkap kata yang sebenarnya. Permainan ini menjadi populer sebagai permainan kata yang menghibur di rumah atau dalam publikasi seperti buku teka-teki dan majalah.

B. Rumusan Masalah

1. Bagaimana cara mengimplementasikan mekanisme pemilihan kata acak tanpa pengulangan?
2. Bagaimana cara menyajikan representasi visual dari kata yang belum ditebak kepada pemain?
3. Bagaimana cara memvalidasi input pemain untuk memastikan hanya huruf tunggal yang valid yang diterima?
4. Bagaimana cara memeriksa kebenaran tebakan pemain dan memperbarui representasi visual kata yang belum ditebak?

C. Tujuan Dan Manfaat

Tujuan

1. Untuk mengetahui bagaimana cara membuat suatu program yaitu game Hangman menggunakan bahasa pemrograman Python.
2. Untuk memenuhi tugas dari mata kuliah proyek berorientasi objek.
3. Untuk mengasah keterampilan pemrograman dan mempelajari konsep dasar dalam pengembangan permainan.

Manfaat

- Pembelajaran bahasa pemrograman: Membuat game Hangman menggunakan Python membantu dalam mempelajari dan memahami sintaks dan konsep dasar dalam bahasa pemrograman Python. Anda dapat mempraktikkan penggunaan variabel, loop, kondisional, fungsi, dan struktur data dalam konteks yang nyata.
- Pemahaman algoritma dan logika pemrograman: Dalam mengembangkan game Hangman, Anda perlu merencanakan algoritma untuk memilih kata secara acak, memvalidasi input pemain, memeriksa kebenaran tebakan, dan mengelola kesempatan pemain.

BAB II

LANDASAN TEORI

1.1 PBO (Pemrograman Berorientasi Objek)

Pemrograman Berorientasi Objek (PBO) adalah paradigma pemrograman yang berfokus pada konsep objek sebagai elemen dasar dalam pengembangan perangkat lunak. Dalam PBO, program dikonstruksi menggunakan objek yang merepresentasikan entitas atau konsep dalam dunia nyata. Setiap objek memiliki atribut (data) dan metode (fungsi) yang memanipulasi data tersebut. Salah satu prinsip utama dalam PBO adalah enkapsulasi, di mana data dan metode yang beroperasi pada data tertentu dikemas bersama dalam satu entitas. Hal ini membantu memisahkan implementasi dari penggunaan objek, meningkatkan modularitas, dan memudahkan pemeliharaan kode.

Konsep berikutnya adalah pewarisan (inheritance), di mana objek dapat mewarisi atribut dan metode dari objek lain, memungkinkan penggunaan kembali kode dan mempercepat pengembangan. Polimorfisme adalah fitur lainnya yang memungkinkan objek dengan jenis yang berbeda untuk merespon metode dengan cara yang berbeda. Selain itu, PBO juga melibatkan penggunaan kelas, yang merupakan blueprint untuk menciptakan objek. Suatu kelas dapat memiliki banyak objek instansiasi yang mewakili berbagai entitas dalam program. Konsep terakhir adalah abstraksi, di mana hanya fitur penting dari suatu objek yang diambil dan ditampilkan, sementara detail implementasinya disembunyikan. Dengan menggunakan PBO, pengembang dapat mengorganisir dan mengelola kode dengan lebih efisien, meningkatkan ketahanan terhadap perubahan, dan memfasilitasi pengembangan perangkat lunak yang lebih kompleks dan mudah dimengerti. Paradigma ini telah menjadi pendekatan yang populer dalam pengembangan perangkat lunak modern dan terus memberikan kontribusi signifikan dalam meningkatkan kualitas serta efisiensi.

1.2 Game Hangman

Hangman adalah permainan tebak kata yang populer dan seru. Tujuan dari permainan ini adalah menebak sebuah kata dengan menebak huruf-huruf yang terdapat dalam kata tersebut. Setiap tebakan yang salah akan mengakibatkan

gambar gantung terisi sedikit demi sedikit, dan jika gambar gantung tersebut lengkap sebelum kata ditebak, pemain kalah. Game Hangman sering dimainkan untuk menguji kosakata dan kemampuan tebak-tebakan seseorang. Selain itu, game ini juga dapat digunakan untuk melatih pemrograman dan logika pemain dalam menerapkan algoritma pemrosesan kata.

Dalam versi modern, permainan Hangman sering dimainkan dengan seorang pemain yang memilih kata secara acak, sedangkan pemain lain harus menebak kata tersebut dengan menebak huruf-hurufnya. Jika pemain yang menebak gagal menebak kata dengan benar sebelum gambar gantungan lengkap tergambar, maka pemain yang memilih kata akan memenangkan permainan. Game Hangman telah berkembang menjadi permainan komputer dan aplikasi seluler yang populer. Ini adalah permainan yang menyenangkan dan menantang yang tetap menjadi favorit banyak orang untuk mengasah keterampilan kosakata dan menguji daya pikir.

Dalam bahasa pemrograman Python, Hangman dapat diimplementasikan dengan memanfaatkan struktur data, pengulangan, dan kondisional. Berikut adalah langkah-langkah umum untuk membuat permainan Hangman menggunakan Python:

Permainan Hangman sering digunakan sebagai cara yang menyenangkan untuk meningkatkan kosakata dan pemahaman terhadap kata-kata. Sederhana dalam konsepnya, permainan ini tetap menjadi favorit di kalangan anak-anak dan orang dewasa. Beberapa variasi dari permainan ini melibatkan aturan khusus atau penambahan elemen desain untuk meningkatkan daya tarik permainan. Meskipun sederhana, Hangman tetap menjadi permainan kata yang populer dan dapat dimainkan oleh berbagai kelompok usia.

Dalam pengembangan game Hangman menggunakan bahasa pemrograman Python meliputi beberapa konsep dan elemen yang perlu dipahami. Berikut adalah beberapa landasan teori yang relevan:

1. Mengambil Kata secara Acak: Anda perlu menggunakan modul `random` dalam Python untuk memilih kata secara acak dari kumpulan kata yang tersedia. Dengan menggunakan fungsi `random.choice()`, Anda dapat memilih kata acak dari daftar kata yang telah ditentukan.

2. Representasi Visual Kata: Dalam permainan Hangman, Anda perlu menampilkan representasi visual dari kata yang belum ditebak. Representasi ini biasanya menggunakan karakter pengganti, seperti garis bawah (`_`), untuk menggantikan huruf-huruf yang belum ditebak. Anda dapat menggunakan string dan penggantian karakter (string manipulation) untuk menciptakan representasi visual ini.
3. Validasi Input Pemain: Anda perlu memvalidasi input yang dimasukkan oleh pemain, yaitu memastikan bahwa input tersebut adalah huruf tunggal. Anda dapat menggunakan kondisional dan fungsi bawaan Python, seperti ``isalpha()``, untuk memeriksa apakah input pemain adalah huruf.
4. Mengecek Kebenaran Tebakan: Setelah menerima input dari pemain, Anda perlu memeriksa apakah huruf tersebut ada dalam kata yang harus ditebak. Anda dapat menggunakan kondisional dan loop untuk memeriksa kebenaran tebakan pemain dengan membandingkan huruf-huruf dalam kata dengan huruf yang ditebak oleh pemain.
5. Mengurangi Kesempatan Pemain: Setiap tebakan yang salah harus mengurangi jumlah kesempatan pemain. Anda dapat menggunakan variabel untuk melacak jumlah kesempatan yang tersisa dan menguranginya setiap kali tebakan pemain salah. Anda juga perlu menggunakan kondisional untuk memeriksa apakah kesempatan pemain sudah habis.
6. Menampilkan Pesan Kemenangan atau Kekalahan: Setelah permainan selesai, Anda perlu menampilkan pesan yang sesuai berdasarkan hasil permainan. Jika pemain berhasil menebak kata, Anda dapat menampilkan pesan kemenangan. Jika pemain kehabisan kesempatan, Anda dapat menampilkan pesan kekalahan.
7. Loop Permainan: Game Hangman biasanya berjalan dalam loop, memungkinkan pemain untuk terus menebak sampai mereka memenangkan permainan atau kehilangan semua kesempatan. Anda dapat menggunakan loop `while` atau loop `for` untuk mengatur jalannya permainan. Dengan memahami landasan teori ini, Kita dapat mengimplementasikan game Hangman yang berfungsi dengan baik menggunakan bahasa pemrograman Python.

BAB III

PEMBAHASAN

A. Source Code

```
import random

class HangmanGame:

    def __init__(self):

        self.words = ["python", "hangman", "programming", "computer", "gaming",
"developer"]

        self.word_to_guess = ""

        self.guessed_letters = []

        self.attempts = 6

    def choose_word(self):

        self.word_to_guess = random.choice(self.words)

    def display_word(self):

        display = ""

        for letter in self.word_to_guess:

            if letter in self.guessed_letters:

                display += letter

            else:

                display += "_"

        return display

    def play(self):

        print("Selamat datang di permainan Hangman!")

        self.choose_word()

        while True:

            current_display = self.display_word()

            print("\nKata yang harus ditebak: " + current_display)
```

```

print("\nSelamat! Kamu berhasil menebak kata: {}".format(self.word_to_guess))

        break

    if self.attempts == 0:
        print("\nMaaf, kamu kalah. Kata yang benar adalah:
{}".format(self.word_to_guess))

        break
    guess = input("\nTebak satu huruf: ").lower()

    if len(guess) != 1 or not guess.isalpha():
        print("Masukkan hanya satu huruf yang valid.")

        continue

    if guess in self.guessed_letters:
        print("Kamu sudah menebak huruf ini sebelumnya. Coba lagi.")

        continue
    self.guessed_letters.append(guess)
    if guess not in self.word_to_guess:
        print("Huruf {} tidak ada dalam kata. Coba lagi.".format(guess))

        self.attempts -= 1

if __name__ == "__main__":

    game = HangmanGame()

    game.play()

```

B. ANALISIS PROGRAM

```
import random

class HangmanGame:
    def __init__(self):
        self.words = ["python", "hangman", "programming", "computer", "gaming", "developer"]
        self.word_to_guess = ""
        self.guessed_letters = []
        self.attempts = 6

    def choose_word(self):
        self.word_to_guess = random.choice(self.words)

    def display_word(self):
        display = ""
        for letter in self.word_to_guess:
            if letter in self.guessed_letters:
                display += letter
            else:
                display += "_"
        return display

    def play(self):
        print("Selamat datang di permainan Hangman!")

        self.choose_word()

        while True:
            current_display = self.display_word()
            print("\nKata yang harus ditebak: " + current_display)
            print("Huruf yang sudah ditebak: " + ", ".join(self.guessed_letters))
            print("Sisa percobaan: {}".format(self.attempts))

            if current_display == self.word_to_guess:
                print("\nSelamat! Kamu berhasil menebak kata: {}".format(self.word_to_guess))
                break

            if self.attempts == 0:
                print("\nMaaf, kamu kalah. Kata yang benar adalah: {}".format(self.word_to_guess))
                break

            guess = input("\nTebak satu huruf: ").lower()

            if len(guess) != 1 or not guess.isalpha():
                print("Masukkan hanya satu huruf yang valid.")
                continue

            if guess in self.guessed_letters:
                print("Kamu sudah menebak huruf ini sebelumnya. Coba lagi.")
                continue

            self.guessed_letters.append(guess)

            if guess not in self.word_to_guess:
                print("Huruf {} tidak ada dalam kata. Coba lagi.".format(guess))
                self.attempts -= 1

if __name__ == "__main__":
    game = HangmanGame()
    game.play()
```

Gambar 3. 1 Source code game hangman

Penjelasan:

Dalam program ini, kami telah mengimplementasikan sebuah permainan "Hangman" menggunakan Python. Kami telah menginisialisasi daftar kata yang dapat dipilih untuk dibakar dan telah mendefinisikan beberapa atribut utama untuk game, seperti kata yang harus ditebak, huruf yang telah ditebak, sisa percobaan, dan status permainan.

Kami telah mendefinisikan beberapa metode dalam class HangmanGame untuk melakukan tugas-tugas yang berbeda dalam permainan ini. Contohnya, metode choose_word() mengatur pemilihan kata yang akan ditebak secara acak dari daftar kata yang telah didefinisikan. Metode display_word() menggantikan huruf

yang belum ditebak dalam kata yang sedang ditebak dengan tanda "_" yang akan ditampilkan pada layar.

Dalam metode `play()`, kami menggunakan perulangan `while` untuk memantau permainan hingga mencapai akhir, seperti menebak semua huruf yang benar dalam kata yang sedang ditebak atau menjalani semua percobaan yang tersedia tanpa menebak semua huruf yang benar. Dalam perulangan ini, kami menampilkan status saat ini dari permainan, meminta input dari pemain, dan menangani kemungkinan percobaan pemain.

Kami telah menjalankan permainan ini secara lokal menggunakan Python, dan permainan bekerja dengan baik dan dapat disajikan pada pemain yang ingin bermain.

Mari kita analisis program nya:

```
import random

class HangmanGame:
    def __init__(self):
        self.words = ["python", "hangman", "programming", "computer", "gaming", "developer"]
        self.word_to_guess = ""
        self.guessed_letters = []
        self.attempts = 6
```

Gambar 3.2 source code class HangmanGame

Penjelasan:

- `import random`: Baris ini mengimpor modul `random`, yang akan digunakan untuk memilih kata secara acak dari daftar kata.
- `class HangmanGame`: Baris ini mendeklarasikan sebuah kelas yang bernama `PermainanHangman` untuk mengelompokkan logika permainan Hangman.
- `def __init__(self)::` Ini adalah metode konstruktor untuk kelas. Ini dieksekusi ketika objek dari kelas dibuat. Parameter `self` merujuk pada instansi kelas, dan di dalam konstruktor, Anda dapat menginisialisasi atribut-atribut kelas.
- `self.kata_kunci = ["python", "hangman", "programming", "komputer", "gaming", "developer"]`: Baris ini membuat sebuah daftar kata yang mungkin harus ditebak oleh pemain. Kata-kata tersebut berhubungan dengan pemrograman dan permainan.

- `self.word_to_guess = ""`: Ini menginisialisasi sebuah string kosong untuk menyimpan kata yang harus ditebak oleh pemain. Nilai ini akan diatur menjadi salah satu kata acak dari daftar `self.kata_kunci` selama permainan.
- `self.guessed_letters = []`: Ini menginisialisasi sebuah daftar kosong untuk menyimpan huruf-huruf yang telah ditebak oleh pemain.
- `self.attempts = 6`: Ini menginisialisasi jumlah upaya yang dimiliki pemain untuk menebak kata. Dalam hal ini, pemain memiliki 6 upaya. Anda dapat menyesuaikan angka ini berdasarkan seberapa sulit Anda ingin permainannya.

```
def choose_word(self):
    self.word_to_guess = random.choice(self.words)

def display_word(self):
    display = ""
    for letter in self.word_to_guess:
        if letter in self.guessed_letters:
            display += letter
        else:
            display += "_"
    return display

def play(self):
    print("Selamat datang di permainan Hangman!")

    self.choose_word()

    while True:
        current_display = self.display_word()
        print("\nKata yang harus ditebak: " + current_display)
        print("Huruf yang sudah ditebak: " + ", ".join(self.guessed_letters))
        print("Sisa percobaan: {}".format(self.attempts))

        if current_display == self.word_to_guess:
            print("\nSelamat! Kamu berhasil menebak kata: {}".format(self.word_to_guess))
            break
```

Gambar 3.3 Potongan Source code game haangman

Penjelasan:

Kode ini adalah program permainan Hangman yang dimainkan melalui konsol. Pemain akan menebak kata yang telah ditetapkan oleh program, dengan mengguess satu huruf setiap waktu. Jika huruf yang dipilih pemain terdapat dalam kata yang harus ditebak, maka huruf tersebut akan ditampilkan. Pemain hanya memiliki jumlah percobaan tertentu, jika pemain menggunakan semua percobaan, maka pemain akan kalah.

Method-method penting yang ada di dalam program ini adalah:

- `select_random_word()`: Memilih secara acak kata dari list kata dan menetapkannya sebagai kata yang harus ditebak.

- `display_word()`: Mengembalikan string yang menunjukkan kata yang harus ditebak dengan huruf yang telah ditebak menggunakan garis bawah.
- `play()`: Melakukan permainan. Ini termasuk menampilkan status saat ini dari permainan, meminta pemain untuk memilih huruf, memeriksa apakah pemain menang atau kalah, dan melanjutkan permainan sampai pemain menang atau kalah.

Di dalam while loop, program akan melakukan hal berikut secara berulang-ulang:

- Memeriksa kata yang harus ditebak saat ini.
- Meminta pemain untuk memilih huruf yang mereka pikirkan termasuk dalam kata yang harus ditebak.
- Memeriksa apakah huruf yang dipilih pemain termasuk dalam kata yang harus ditebak. Jika iya, maka huruf tersebut akan ditampilkan dan ditaruh dalam list huruf yang telah ditebak.
- Memeriksa apakah pemain telah menebak semua huruf yang ada dalam kata yang harus ditebak. Jika iya, maka pemain menang.
- Memeriksa apakah pemain telah menggunakan semua percobaan yang ada. Jika iya, maka pemain kalah.

Kode ini dapat digunakan oleh pemula yang ingin belajar cara mengimplementasikan algoritma pencarian kata menggunakan Python. Kode ini juga dapat digunakan sebagai dasar untuk memperluas fungsi permainan, seperti menambahkan dukungan untuk kata dengan makna, menambahkan waktu tunggu antar percobaan, dan sebagainya.

Pemrogram dapat menggali konsep-konsep lanjutan seperti penanganan kata-kata dengan makna atau topik khusus, implementasi antarmuka grafis pengguna untuk meningkatkan aspek visual permainan, dan penambahan elemen permainan lainnya seperti sistem skor. Dengan mengeksplorasi dan memperluas fungsionalitas permainan Hangman, pemula dapat memperdalam pemahaman mereka tentang bahasa pemrograman Python sambil menciptakan proyek yang lebih kompleks dan menarik.

```

        if self.attempts == 0:
            print("\nMaaf, kamu kalah. Kata yang benar adalah: {}".format(self.word_to_guess))
            break

        guess = input("\nTebak satu huruf: ").lower()

        if len(guess) != 1 or not guess.isalpha():
            print("Masukkan hanya satu huruf yang valid.")
            continue

        if guess in self.guessed_letters:
            print("Kamu sudah menebak huruf ini sebelumnya. Coba lagi.")
            continue

        self.guessed_letters.append(guess)

        if guess not in self.word_to_guess:
            print("Huruf {} tidak ada dalam kata. Coba lagi.".format(guess))
            self.attempts -= 1

if __name__ == "__main__":
    game = HangmanGame()
    game.play()

```

Gambar 3.4 Potongan Source code game haangman

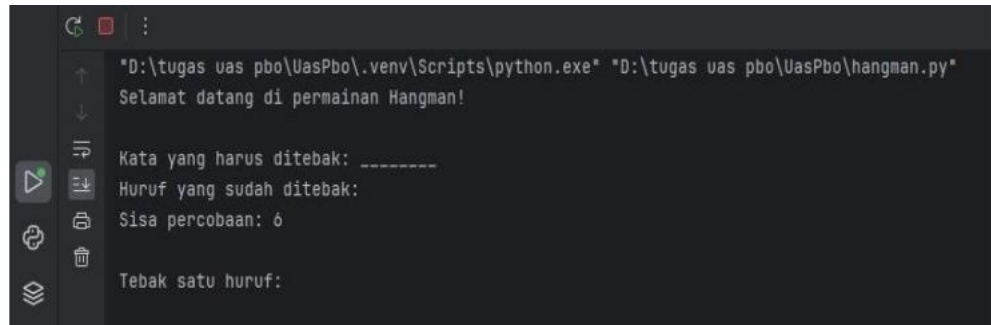
Penjelasan :

Kode ini merupakan bagian dari permainan Hangman yang melibatkan input dari pengguna dan logika pemrosesan tebakan. Mari kita bahas dengan jelas dan sederhana:

- *guess = input("\nTebak satu huruf: ").lower()* : Baris ini meminta pengguna untuk memasukkan satu huruf dan menyimpan input tersebut dalam variabel *guess*. Fungsi *lower()* digunakan untuk memastikan bahwa huruf yang dimasukkan akan diubah menjadi huruf kecil agar konsisten.
- *if len(guess) != 1 or not guess.isalpha(): print("Masukkan hanya satu huruf yang valid.") continue* : Ini adalah pengujian kondisional yang memeriksa apakah input pengguna hanya satu huruf dan merupakan huruf alfabet. Jika tidak, pesan kesalahan dicetak dan program melanjutkan kembali ke langkah input.
- *if guess in self.guessed_letters: print("Kamu sudah menebak huruf ini sebelumnya. Coba lagi.") continue* : Pernyataan ini memeriksa apakah huruf yang ditebak oleh pengguna sudah pernah ditebak sebelumnya. Jika iya, pesan kesalahan dicetak dan program kembali ke langkah input.
- *self.guessed_letters.append(guess)* : Jika huruf belum pernah ditebak sebelumnya, maka huruf tersebut ditambahkan ke daftar *guessed_letters*.

- *if guess not in self.word_to_guess: print("Huruf {} tidak ada dalam kata. Coba lagi.".format(guess)) self.attempts -= 1*: Pernyataan ini memeriksa apakah huruf yang ditebak tidak ada dalam kata yang harus ditebak (word_to_guess). Jika iya, pesan kesalahan dicetak, dan jumlah upaya (attempts) dikurangi satu

C. Penjelasan OUTPUT



```
"D:\tugas uas pbo\UasPbo\.venv\Scripts\python.exe" "D:\tugas uas pbo\UasPbo\hangman.py"
Selamat datang di permainan Hangman!

Kata yang harus ditebak: _____
Huruf yang sudah ditebak:
Sisa percobaan: 6
Tebak satu huruf:
```

Gambar 3.5 Output game Hangman

Penjelasan :

Output yang dihasilkan akan berupa pesan-pesan yang muncul selama permainan, satu kata yang sedang ditebak, jumlah kesempatan yang tersisa, serta representasi visual dari state hangman yang berubah seiring dengan permainan berlangsung.

Berikut merupakan penjelasan luaran tersebut :

- "Selamat datang di permainan Hangman!": Ini adalah pesan selamat datang yang dicetak saat permainan dimulai.
- "Kata yang harus ditebak: _____": Ini adalah tampilan awal dari kata yang harus ditebak. Setiap huruf dalam kata tersebut direpresentasikan oleh garis bawah (underscore), yang berarti belum ada huruf yang ditebak.
- "Huruf yang sudah ditebak: ": Saat awal permainan, belum ada huruf yang ditebak, jadi bagian ini kosong.
- "Sisa percobaan: 6": Ini menunjukkan bahwa pemain memiliki 6 percobaan (kesalahan) untuk menebak kata tersebut.
- "Tebak satu huruf: ": Ini adalah prompt yang mengajak pemain untuk memasukkan tebakan satu huruf.

Dengan kata lain, pemain diminta untuk menebak huruf satu per satu. Setelah pemain memasukkan huruf pertama, permainan akan memberikan informasi lebih lanjut, seperti huruf yang sudah ditebak, tampilan kata yang terkini, dan sisa percobaan. Selanjutnya, pemain dapat melanjutkan dengan menebak huruf-huruf lain atau mengakhiri permainan.

```
Tebak satu huruf: e

Kata yang harus ditebak: deve__e_
Huruf yang sudah ditebak: y, m, v, d, e
Sisa percobaan: 4

Tebak satu huruf: l

Kata yang harus ditebak: devel_e_
Huruf yang sudah ditebak: y, m, v, d, e, l
Sisa percobaan: 4

Tebak satu huruf: o

Kata yang harus ditebak: develo_e_
Huruf yang sudah ditebak: y, m, v, d, e, l, o
Sisa percobaan: 4

Tebak satu huruf: p

Kata yang harus ditebak: develop_e_
Huruf yang sudah ditebak: y, m, v, d, e, l, o, p
Sisa percobaan: 4

Tebak satu huruf: r

Kata yang harus ditebak: developer
Huruf yang sudah ditebak: y, m, v, d, e, l, o, p, r
Sisa percobaan: 4

Selamat! Kamu berhasil menebak kata: developer
```

Gambar 3.6 Output Percobaan game Hangman

Penjelasan:

Dalam permainan Hangman ini, pemain diminta menebak sebuah kata yang dipilih secara acak, dan setiap huruf dari kata tersebut disembunyikan. Pemain dapat menebak satu huruf pada setiap tahap, dan hasilnya ditampilkan, dengan huruf yang telah benar ditebak diungkap. Jika huruf yang ditebak tidak ada dalam kata, pemain kehilangan satu percobaan. Pemain dapat melihat kata yang harus ditebak, huruf yang sudah ditebak, dan sisa percobaan setiap saat. Permainan berlanjut hingga pemain berhasil menebak seluruh kata atau kehabisan percobaan.

Dalam permainan Hangman, pemain dihadapkan dengan tantangan untuk menebak sebuah kata yang telah dipilih secara acak oleh permainan. Setiap huruf dari kata tersebut disembunyikan, dan pemain diberikan kesempatan untuk menebak satu huruf pada setiap tahap permainan. Hasil dari tebakan tersebut kemudian ditampilkan, dengan huruf-huruf yang telah benar ditebak diungkap, sedangkan yang belum tetap disembunyikan.

BAB IV

KESIMPULAN DAN SARAN

A. KESIMPULAN

Dalam laporan ini, telah disimpulkan bahwa konsep pemrograman berorientasi objek (OOP) memberikan pendekatan yang terstruktur dan modular dalam pembuatan game Hangman menggunakan Python. penggunaan konsep OOP dalam pemrograman game Hangman menggunakan Python memberikan manfaat besar. Ini meningkatkan struktur, modularitas, dan keterbacaan kode. Konsep OOP memungkinkan pengembang untuk memecah permasalahan menjadi bagian-bagian yang lebih kecil dan terkelola dengan lebih baik. Selain itu, OOP memfasilitasi pemeliharaan dan pengembangan berkelanjutan dengan memisahkan perubahan pada satu objek tanpa memengaruhi yang lain. Dengan penerapan konsep OOP ini, pengembang dapat menciptakan game Hangman yang lebih fleksibel dan dapat diperluas.

B. SARAN

Semoga Game Hangman yang kami buat ini dapat bermanfaat, khususnya bagi kami. Penulis menyadari bahwa penulis masih jauh dari kata sempurna oleh karena itu, kami sangat mengharapkan kritik dan saran yang bersifat memba

DAFTAR PUSTAKA

Cara Bermain Hangman. TheFastCode. (n.d.). <https://www.thefastcode.com/id-idr/wiki/Bermain-Hangman>

Hangman. html5. (n.d.). <https://www.gamestolearnenglish.com/hangman/>

Herwanto, H. W., & Febrita, R. E. (2015). Pengembangan Media Pembelajaran Berbasis Web pada Matakuliah Pemrograman Berorientasi Objek. *TEKNO*, 21(1).

Pane, S. F., & Saputra, Y. A. (2020). *Big Data: Classification Behavior Menggunakan Python* (Vol. 1). Kreatif.

wikiHow. (2018, July 30). *Cara Bermain Hangman: 11 Langkah (Dengan Gambar)*. wikiHow. <https://id.wikihow.com/Bermain-Hangman>



**KEMENTRIAN PENDIDIKAN, KEBUDAYAAN, RISET,
DAN TEKNOLOGI
UNIVERSITAS BENGKULU
FAKULTAS TEKNIK
PROGRAM STUDI SISTEM INFORMASI**
Jl. Wr. Supratman Kandang Limun, Bengkulu
Bengkulu 38371 A Telp: (0736) 344087, 22105 - 227

**LEMBAR ASISTENSI
UJIAN AKHIR SEMESTER PEMROGRAMAN BERORIENTASI OBJEK**

Nama Mahasiswa : : 1. Esra Silvia Sihite (G1F022035)
2. Muhammad Rifky Ramadhani (G1F022037)
3. Muhammad Salman Alfarizi (G1F022049)

Dosen : 1. Kurnia Anggriani, S.T., M.Eng
2. Arie Vatesia, S.T., M.TI, Ph.D.

Asisten Dosen : : 1. Randi Julian Saputra (G1A019066)

Ujian Akhir Semester	Catatan dan Tanda Tangan
Ujian Akhir Semester	