**PRAKTIKUM PEMROGRAMAN BERORENTASI OBJEK**

**FINAL PROJECT GREENFOOT**

**“**Membuat Game flying-Bird**”**

****

**Dosen Pengampu:**

**Sayekti Harits Suryawan, S.Kom.,M.Kom.**

**Oleh**

1. Nindy miftahur Rizqi (2211102441160)
2. Edwin Abdul Salam (2211102441138)
3. Indah Pariana (2211102441164)
4. Sarina Safitri (221102441218)

**UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR**

**FAKULTAS SAINS DAN TEKNOLOGI**

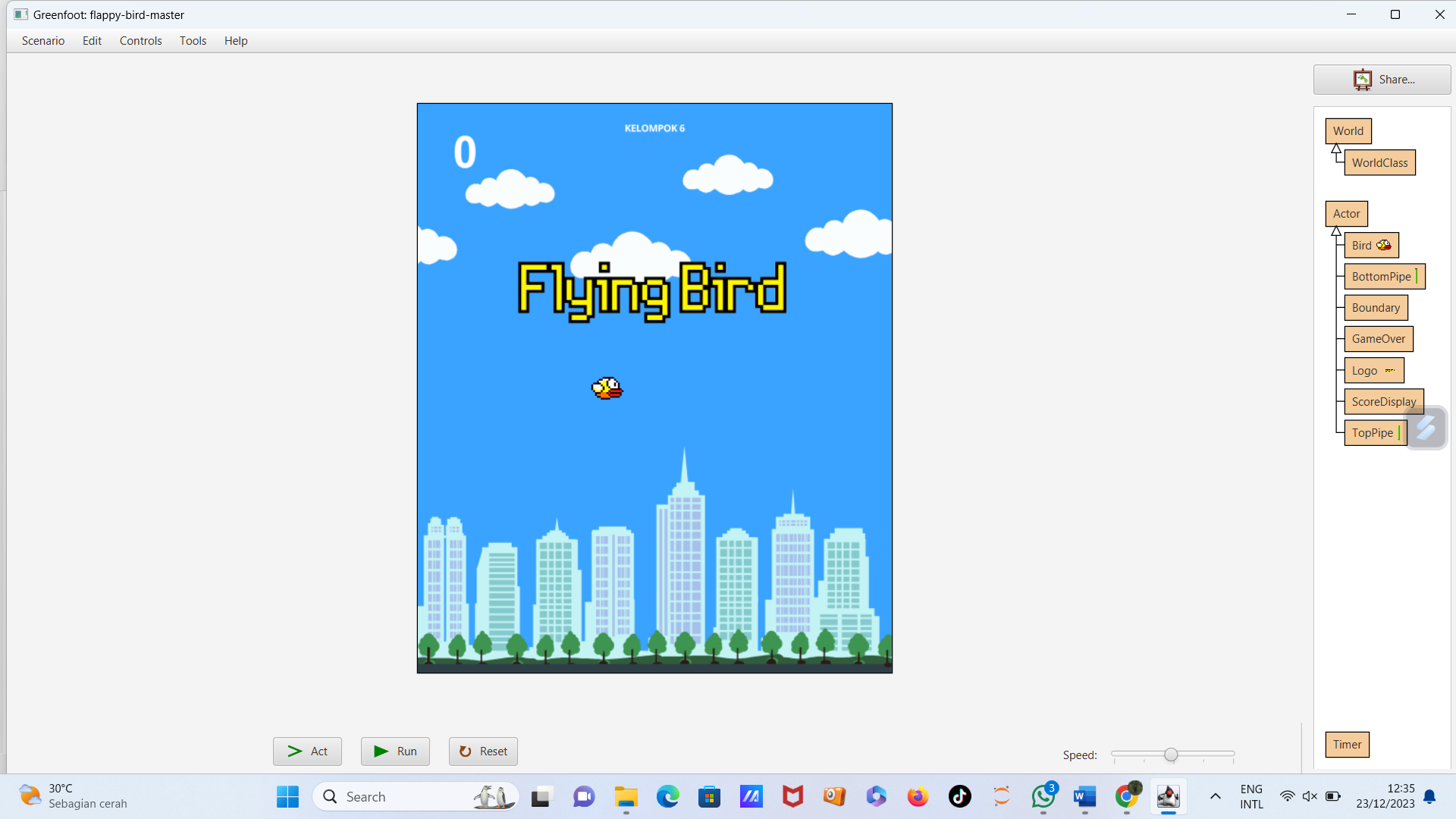
**2023/2024**

1. **Pendahuluan**

Game Flying Bird merupakan salah satu permainan yang populer dan sederhana yang dikembangkan menggunakan platform pemrograman Greenfoot. Dalam pengembangan permainan ini, berbagai konsep dasar dalam pemrograman berorientasi objek diterapkan untuk menciptakan pengalaman bermain yang menyenangkan dan menarik.

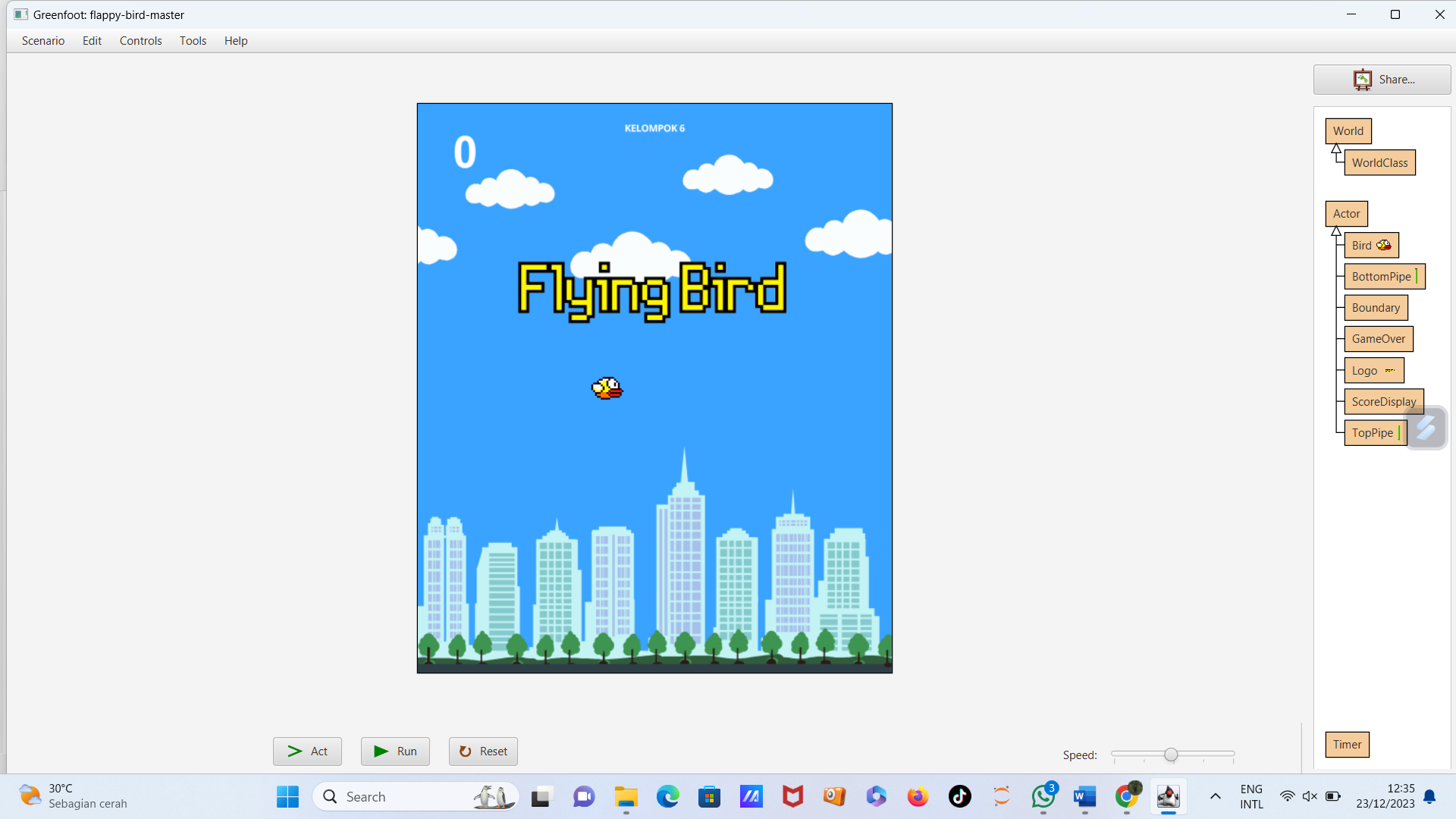
Dokumen ini akan memberikan gambaran umum tentang struktur dan implementasi kode dalam proyek Flying Bird. Konsep-konsep utama seperti inheritance, polymorphism, overloading, dan overriding akan dibahas secara rinci untuk memberikan pemahaman mendalam tentang cara permainan ini dirancang dan diimplementasikan.

Berikut gambaran serta isi pada game Flying Bird

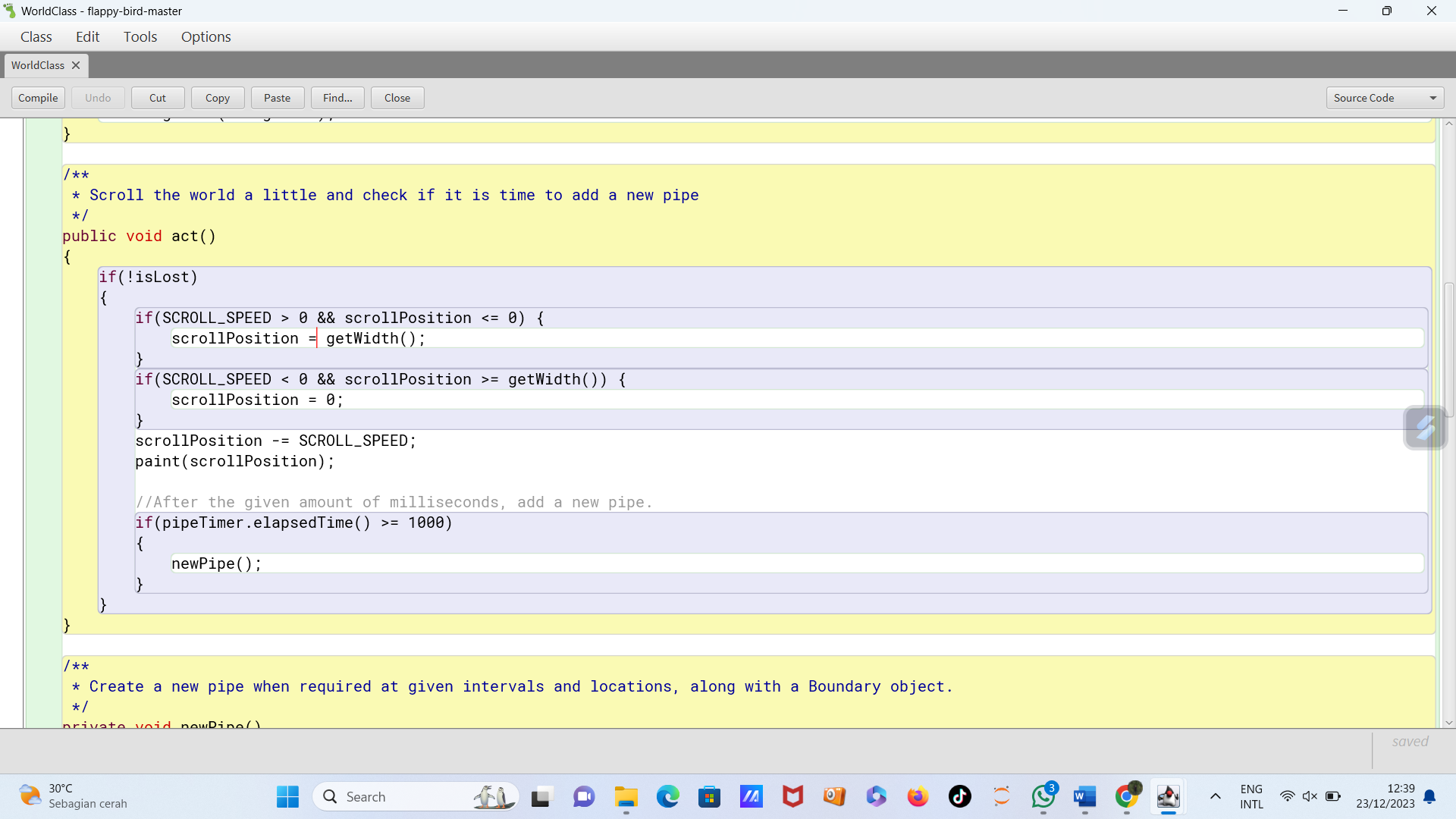


1. **Kelas WorldClass**kelas WorldClass adalah kelas utama yang mengatur dunia permainan. Berikut adalah beberapa poin penting dari kelas ini :

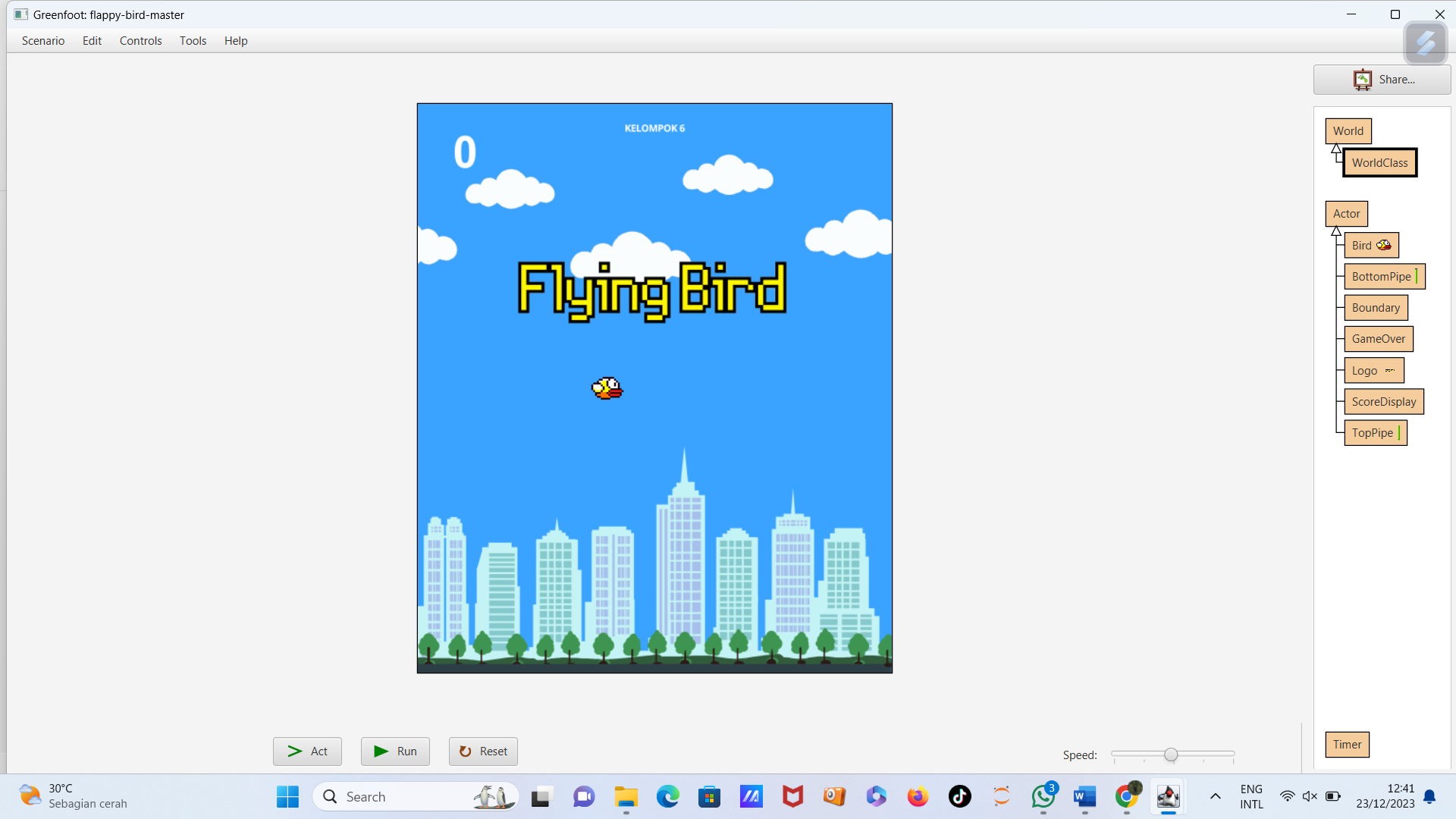
* **Inheritance : ‘WorldClass’**  mewarisi sifat dan metode dari kelas ‘World’ dalam Greenfoot. Ini memungkinkan pembuatan dunia permainan dengan karakteristik khusus untukk permainan Flying Bird.



* **Overriding : metode ‘act’**  di-override untuk mengontrol pergerakan dunia dan menambahkan pipa baru setiap beberapa detik.



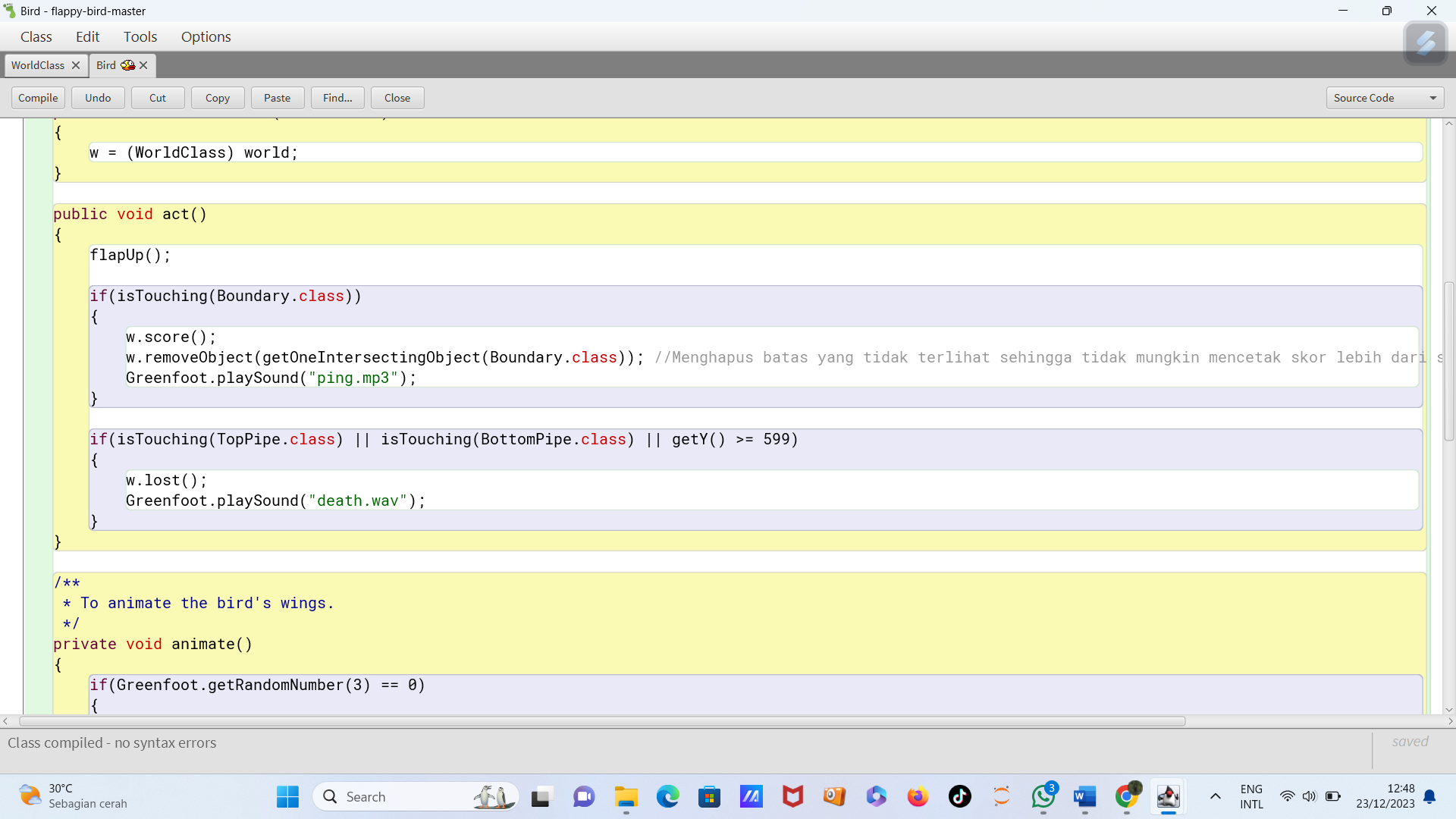
* **Polymorphism :**  berbagai objek seperti burung, pipa, dan skor diatur dalam dunia sebagai objek-objek kelas **‘Actor’**.



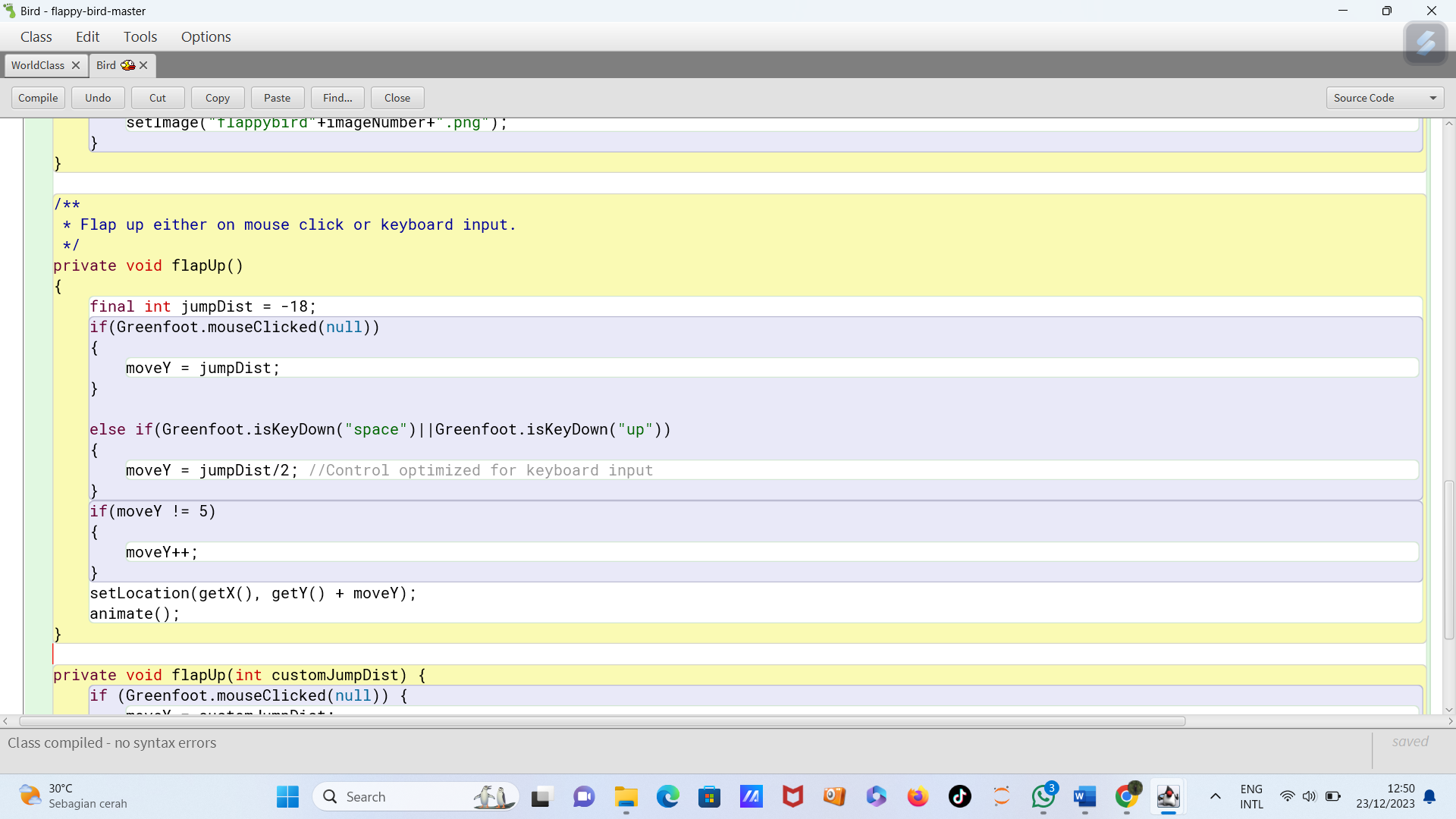
1. **Kelas Bird**

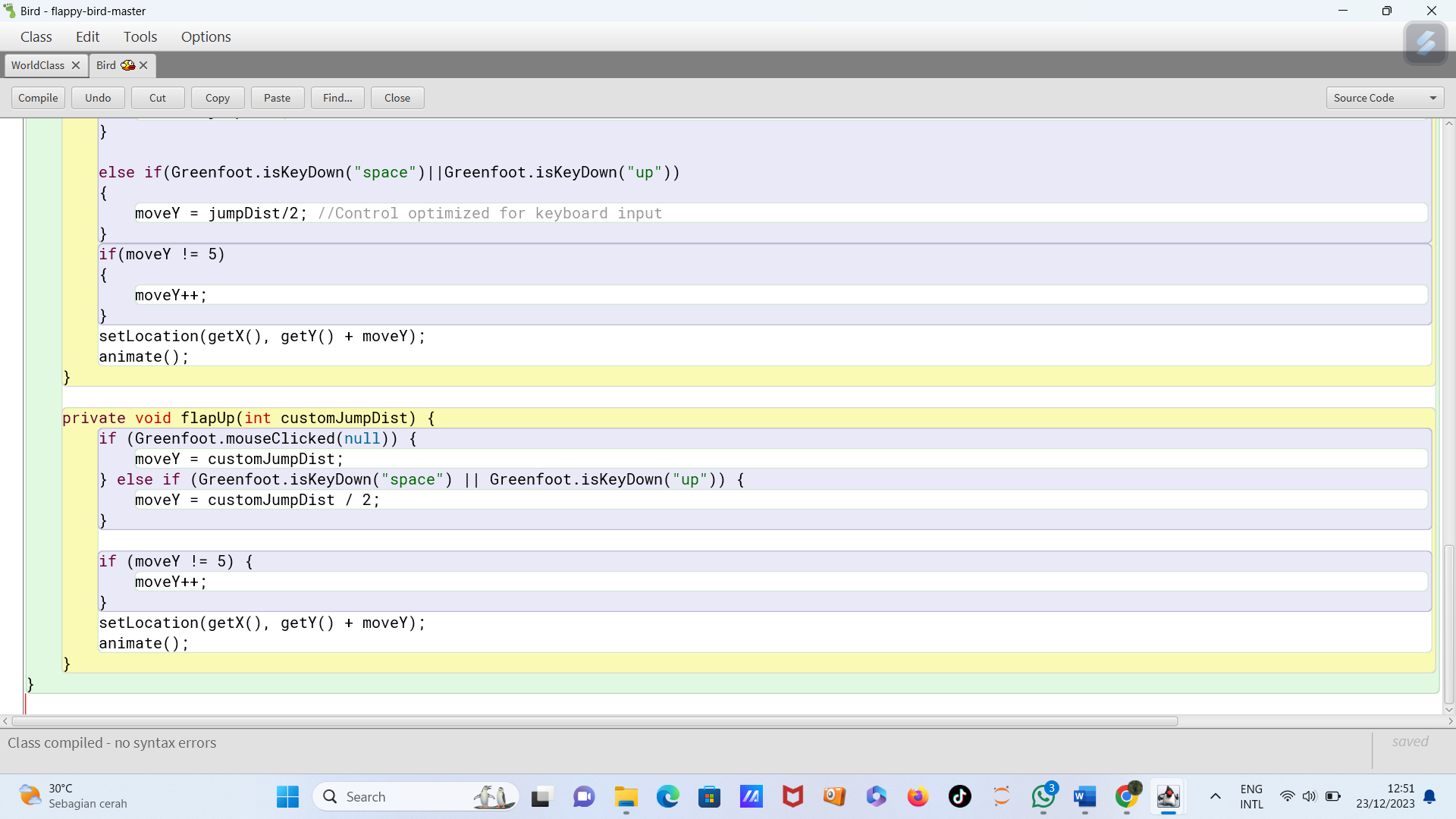
Kelas ‘**Bird’** mewakili karakter utama, yaitu burung yang dikendalikan oleh pemain. Beberapa aspek kelas ini:

* **Overriding:** Metode **‘act’** dan animate di-overide dari kelas Actor. Ini memungkinkan implementasi perilaku khusus untuk burung, termasuk aksi ketika berinteraksi dengan pipa dan deteksi tabrakan.



* **Overloading:** Terdapat dua versi dari metode flapUp. Salah satu versi tanpa parameter dan yang lainnya dengan parameter customJumpDist. Ini memberikan fleksibilitas saat burung melompat, tergantung pada input pengguna.

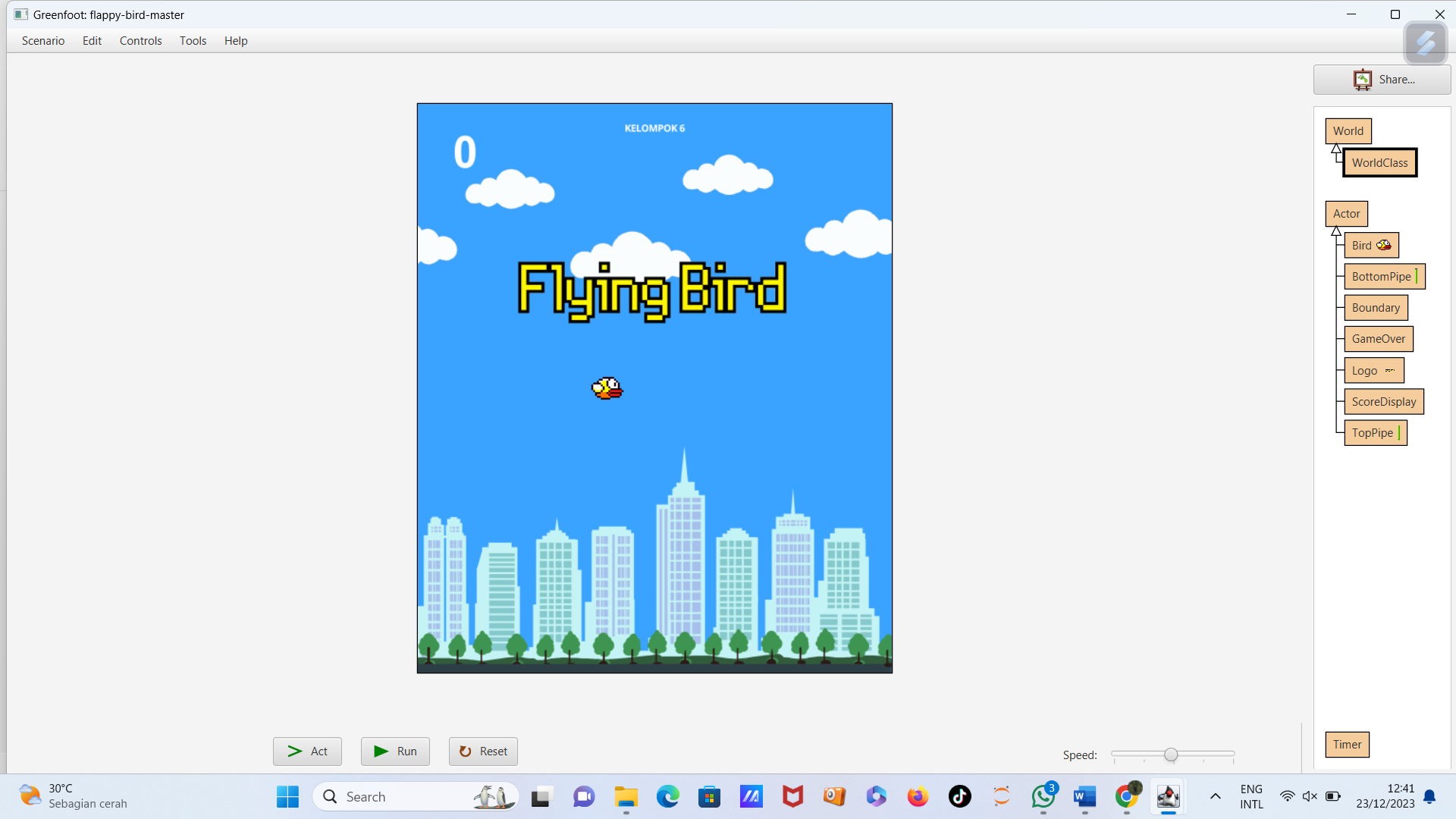




1. **Kelas TopPipe dan BottomPipe**

Kelas **‘TopPipe’** dan **‘BottomPipe’** merepresentasikan bagian atas dan bawah dari pipa di permainan. Poin-poin penting:

* **Inheritance:** Kedua kelas ini mewarisi sifat dan metode dari kelas **‘Actor’**.



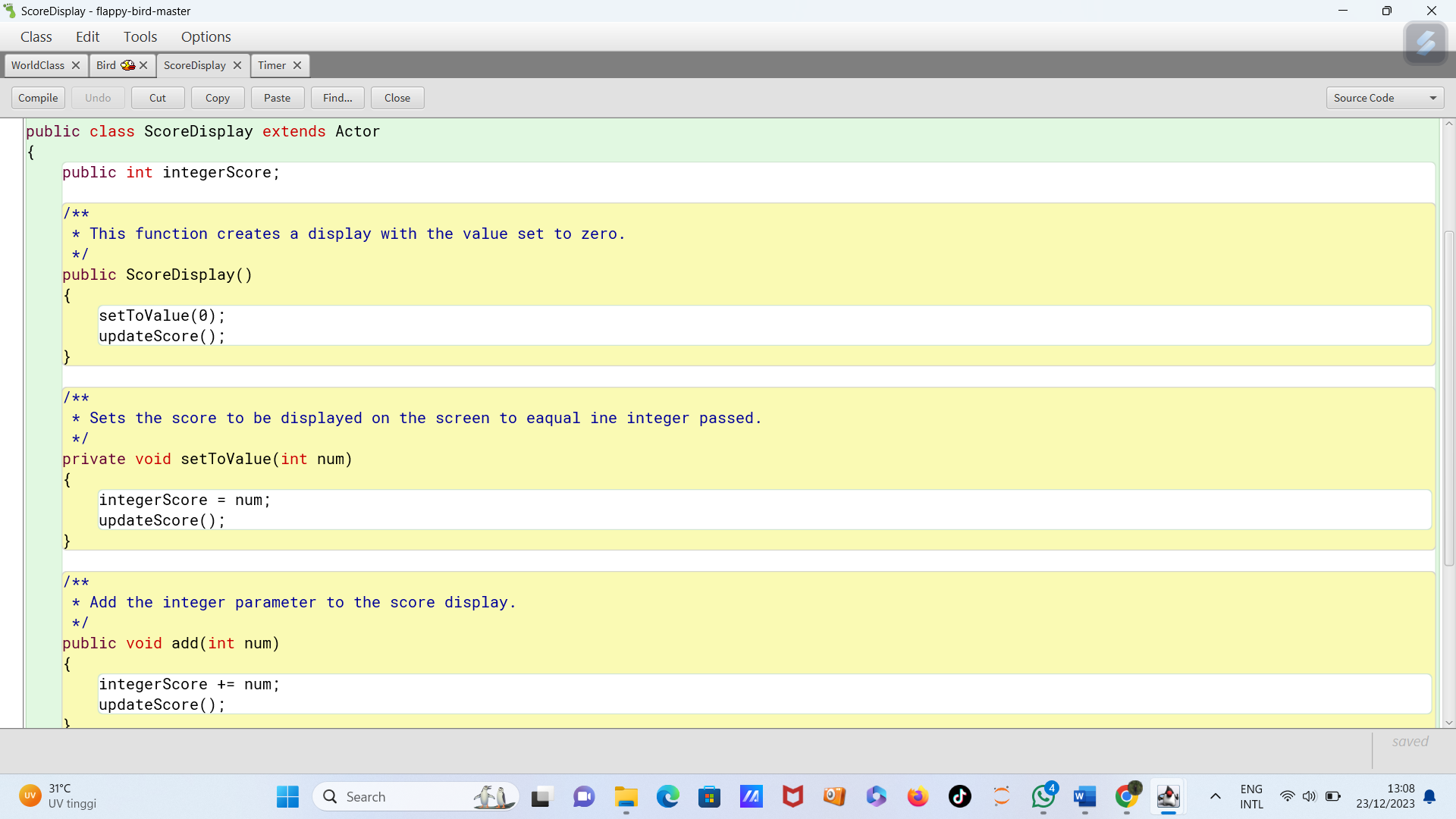


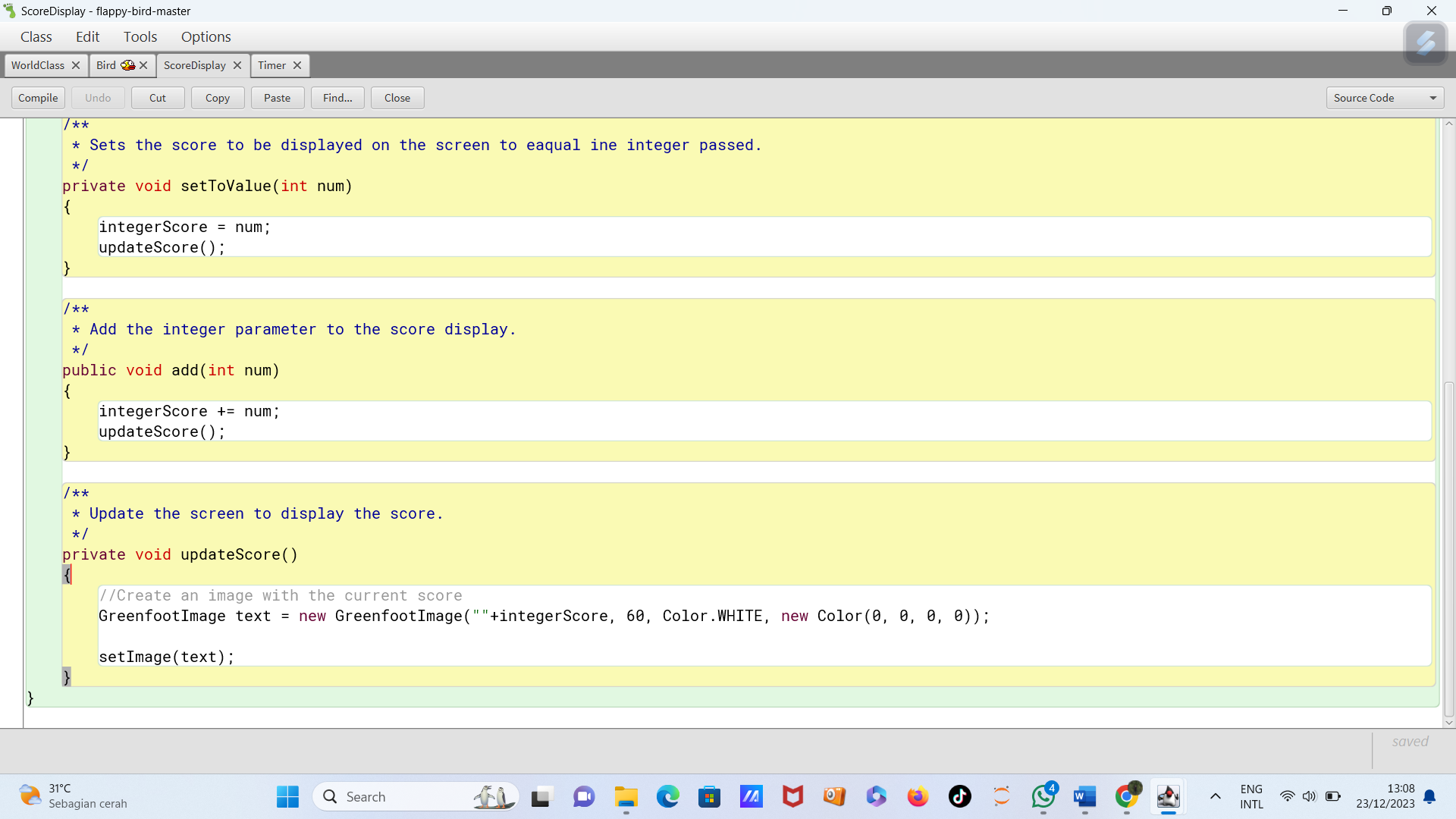
* **Polymorphism:** Objek-objek ini dapat ditempatkan di dunia sebagai objek-objek **‘Actor’**.

1. **Kelas ScoreDisplay**

Kelas ScoreDisplay bertanggung jawab untuk menampilkan skor permainan sebagai teks pada layar. Beberapa hal penting:

* **Overloading:** Terdapat metode add yang memungkinkan penambahan nilai ke skor saat ini. Metode ini memberikan fleksibilitas dalam menentukan jumlah skor yang ingin ditambahkan.
* **Encapsulation:** Variabel integerScore diakses melalui metode setToValue dan add, memastikan pengaturan nilai skor melalui metode-metode ini untuk menjaga integritas data.
* **Update Tampilan:** Metode updateScore menciptakan objek GreenfootImage baru dengan teks yang menggambarkan nilai skor saat ini. Teks memiliki ukuran 60, warna putih, dan latar belakang transparan. Gambar tersebut kemudian diatur sebagai gambar kelas ScoreDisplay.

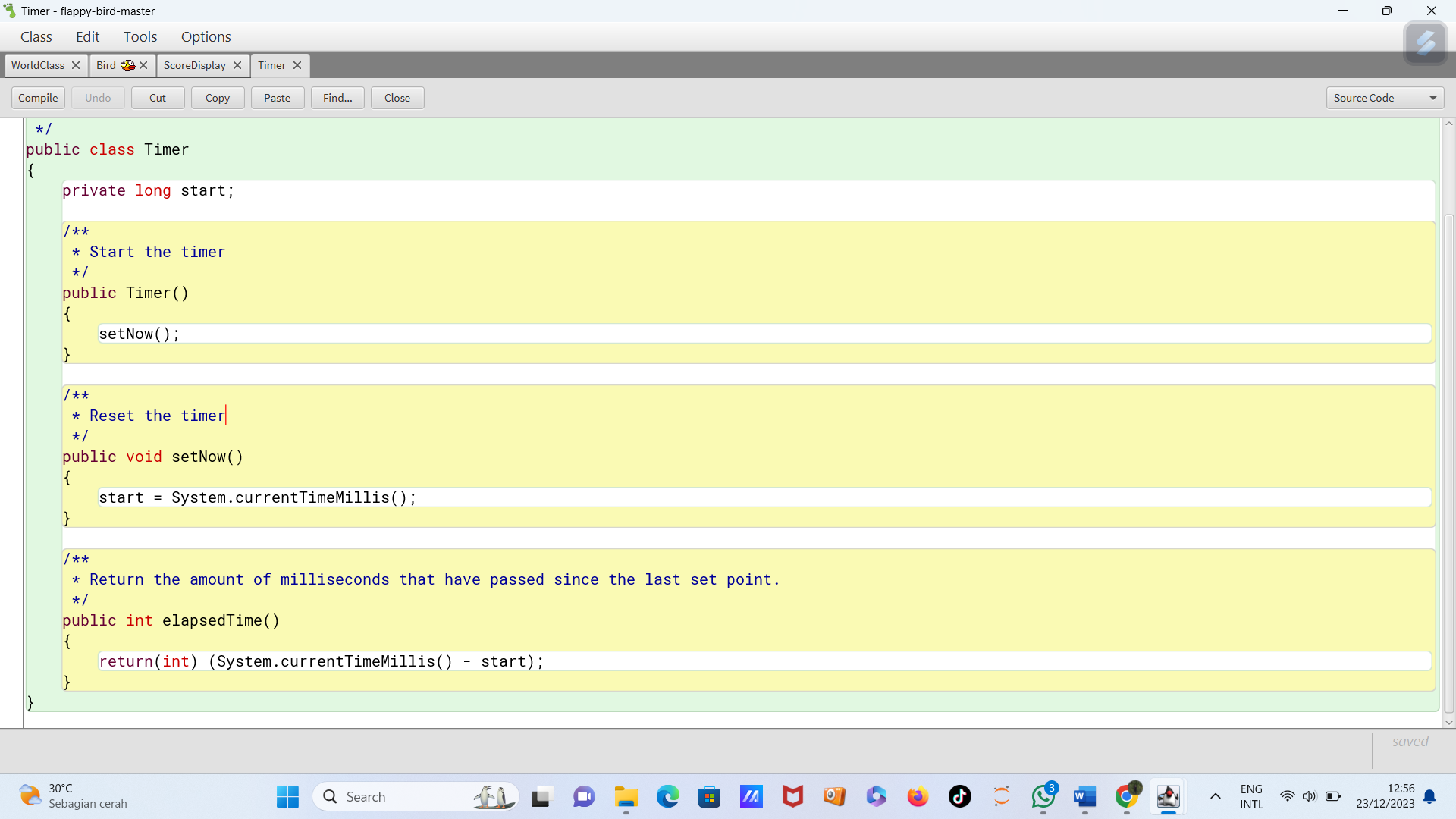
Berikut adalah isi kode dari Kelas ScoreDisplay  




1. **Kelas Timer**

Kelas **‘Timer’** adalah implementasi sederhana dari timer dalam permainan. Poin pentingnya :

* **Encapsulation:** Menggunakan variabel **‘start’** sebagai variabel instance privat.



* **Polymorphism:** Meskipun tidak terlibat dalam polimorfisme, tetapi konsepnya dapat diaplikasikan di tempat lain dengan membuat turunan dari kelas ini.

1. **Kesimpulan**Pengembangan permainan **"Flying Bird"** menggunakan platform Greenfoot memperlihatkan penerapan konsep-konsep dasar dalam pemrograman berorientasi objek. Dengan menggunakan inheritance, kelas-kelas seperti **`WorldClass`**, **`TopPipe`**, dan **`BottomPipe`** dapat mewarisi sifat dan metode dari kelas-kelas yang ada, memberikan struktur yang konsisten dalam pembentukan dunia permainan. Overriding digunakan pada kelas **`Bird`** untuk mengimplementasikan perilaku khusus, seperti interaksi dengan pipa dan deteksi tabrakan. Polymorphism terlihat melalui objek-objek kelas **`Actor`**, yang dapat ditempatkan di dunia dan diatur dengan tipe data kelas induk, meningkatkan fleksibilitas dalam penggunaan. Overloading diimplementasikan pada kelas **`Bird`** dengan dua versi metode **`flapUp`**, memberikan kemampuan untuk lompatan burung dengan atau tanpa parameter tambahan. Terakhir, konsep enkapsulasi digunakan dalam kelas **`Timer`**, melindungi variabel **`start`** dari akses langsung. Secara keseluruhan, penggunaan konsep-konsep ini memperkuat struktur dan keterbacaan kode, memudahkan pengelolaan, dan memberikan wawasan praktis tentang penerapan pemrograman berorientasi objek dalam pengembangan permainan.