Inheritance Game of Thrones

Welcome to the **Inheritance Game!** We'll use a **fantasy world inspired by Game of Thrones** to practice object-oriented programming concepts in Java.

Don't worry if you haven't watched the show or read the books, you don't need any prior knowledge to complete this problem. The fantasy theme is just a fun way to help you remember the relationships, abilities, and behaviors in this problem set.

Setting the Scene

- **Characters** belong to different families (like the Starks or Lannisters) and have unique behaviors.
- **Direwolves** and **dragons** are special creatures with their own abilities.
- Only some creatures can **fly**, so we'll use a **Flyable interface** for them.
- Each character and creature can "speak" in their own way, using the `speak()` method.

Your task is to **fix the hierarchy and implement the missing code**.
<u>Write all the classes and interfaces in one single App.java file.</u>

GIVEN BROKEN DESIGN

```
class Character { }
class JonSnow extends Character { }
class Dragon extends Character { } // WRONG DESIGN
```

You must **fix the design**, **create hierarchy**, and **complete the system**.

TASK 1 — Interface & Design Fix (8 marks)

TODO 1.1 — Create interface

```
// TODO: create interface Flyable
// It has ONE method: fly()
```

TODO 1.2 — Fix Dragon class

- Must **NOT extend Character**
- Must **implement Flyable**
- `fly()` prints: `Dragon flying`

TASK 2 — Character Hierarchy & Visibility

TODO 2.1 — **Character** superclass

Requirements:

- `name` → **private**
- `age` → **protected**
- Constructor accepts (String name, int age) and prints: `Character created`
- **public getter** for `name`
- `speak()` prints `"..."`

```
class Character {
    // TODO
}
```

TODO 2.2 — **Stark** subclass

- Extends `Character`
- Constructor accepts (String name, int age), calls super, prints: `Stark created`

TODO 2.3 — **JonSnow** class

- Extends `Stark`
- Constructor accepts (String name, int age), calls super, prints: `Jon Snow created`
- Override:
    - `speak()` → `"I know nothing"`

TODO 2.4 — **Arya** class

- Extends `Stark`
- Constructor accepts (String name, int age), calls super, prints: `Arya created`
- Override:
    - `speak()` → `"Not today"`

TODO 2.5 — **Lannister & Tyrion**

Lannister:

- Extends `Character`
- Constructor accepts (String name, int age), calls super, prints: `Lannister created`
- `speak()` → `"I always pay my debts"`

Tyrion:

- Extends `Lannister`
- Constructor accepts (String name, int age), calls super, prints: `Tyrion created`
- `speak()` → `"I drink and I know things"`

TODO 2.6 — **Direwolf**

- Extends `Character`
- Constructor accepts (String name, int age), calls super, prints: `Direwolf created`
- `speak()` → `"Howl"`

TASK 3 — DO NOT MODIFY (Main Auto-Tester)

Once you are done, use the provided **Test.java** to test your code. Your code **might be correct if all tests pass**. Put the **Test.java** in the same directory as **App.java** (i.e., in the same package, main).