Taller 7 java

Completado	☐
Cursos	👨‍💻 POO
Fecha de entrega	@15 de enero de 2025
Grupo	1
Persona	Ⓙ Jeronimo Rua Herrera

parte 1

1. Porque es una clase que su función es entregar responsabilidades a los que la heredan, por ejemplo todos los instrumentos deben sonar pero cada instrumento suena distinto, por lo que da la orden de “los instrumentos deben sonar” pero, cada uno que lo haga a su manera. En este caso no hay sonar pero el método tocar es similar al ejemplo que otorgué.
2. Así lo haría yo.

```
class Piano extends Instrumento{
    public Piano(String tipo){super(tipo)}
    public void tocar(){sout("tocando piano")}
    public void afinar(){sout("no afines un piano si no eres
}
```

3. No se si soy yo pero por generalización yo no veo ningún tipo de error.
4. Debería imprimir “tocando saxofón”; “tocando guitarra”.

parte 2

1. Dara un error de compilación, los métodos abstractos no pueden tener definicion, no pueden tener cuerpo.

2. No, no considero ninguno de los dos como un error ya que el ID de un objeto se obtiene igual sin importar desde qué clase esté, mientras se defina de una forma correcta el ID y tampoco considero un error el método tipoCuerpo2 “extra solar” ya que siempre estamos hablando de un objeto extra solar, como el nombre de la clase lo indica.
3. Seria un poco innecesario pero en sin un error no, porque no estamos creando un objeto de este tipo en ninguna parte, simplemente queremos que las darme clases hereden de esto y con el abstrac prevenimos la indeseada creación de un objeto del tipo. Ósea básicamente depende del objetivo.
4. Es sencillamente un uso de generalización, queremos que en el array puedan haber todo tipo de clase que hereda de ObjetoAstronomicoExtraSolar, ya en la linea 25, lo que hacemos es recorrer uno a uno los objetos del array, y por ligadura dinámica, correr su método.
5. Porque en la linea anterior indicamos que ahora el espacio 0 del array va a apuntar a un objeto SuperNova.
6. Porque la clase Estrella también es abstracta, ósea no tiene que cumplir con las clausulas de su padre.
7. Porque un método no puede ser abstracto y private al mismo tiempo.
8. Porque es hija de Estrella mas no de ObjetoAstronomicoExtraSolar, si se define no pasaría nada. Simplemente un método más.
9. Porque hará uso del toString por defecto de el Objeto OBject, padre de todos.
10. Porque se pueden crear punteros de clases abstractas, lo que no se puede es crear objetos de una clase abstracta.
11. la B es invalida, no se puede crear una instancia de una clase abstracta, la C es correcta por generalización y la posibilidad de poder crear punteros de clases abstractas.
12. La instrucción B es correcta por generalización, la C no entendí la pregunta.
13. Porque cualquier objeto es hijo de la clase object.
14. No, esta bien, tiene sentido en el caso de que queramos que nuestras clases hijas usen este constructor, si no no aporta mucho la verdad.

15. El error es que debe crear el método explotar por obligación, en el caso de que no se quiera tener que implementar este método se puede optar por convertirla en una clase abstracta.