Respuestas

1) Ejercicio de analisis

a) La clase instrumento debe definirse como abstracta debido a que utiliza métodos abstractos como los son **Tocar()** y **Afinar()** los cuales no tienen implementación en la clase instrumento, por tanto una clase con al menos un método abstracto debe definirse abstracta

Diferencia con una clase normal:

- No se pueden crear instancias de una clase abstracta, además de que los métodos que se definen como abstractos deben de ser implementados por sus subclases
- Por otro lado, se pueden crear instancias de las clases normales y estas no contienen métodos abstractos.

b) Elaboración Clase Piano

```
public class Piano extends Instrumento {
    public Piano(String Tipo){ super(Tipo); }

    public void Tocar(){ System.out.println("Tocando Piano");}
    public void Afinar(){ System.out.println("Afinando Piano");}
}
```

c) En el siguiente código no hay errores de compilación, aunque Instrumento sea una clase abstracta se puede crear una referencia de tipo Instrumento. Además cuando se asignan instancias de las subclases Saxofon como Guitarra a la referencia abstracta x es valido pues estas son subclases que heredan de Instrumento

```
public class Test {
    public static void main(String[] args) {
        Instrumento x;
        x = new Saxofon("xxxxx");
        x = new Guitarra("xxxxx");
    }
}
```

d) Tocando Saxofon
Tocando Guitarra

2) Ejercicio de código en el repositorio

a) El código genera un error ya que se esta intentando definir un método abstracto, los cuales no tienen un cuerpo/definición especifica

b) Los métodos tipoCuerpo2() y getID() en la clase ObjetoAstronomicoExtrasolar no están definidos como abstractos debido a que tienen un cuerpo/definición

especifica concreto. Esto no se considera un error ya que permite a las subclases heredar estas funcionalidades sin tener que redefinirlas.

c) No se puede considerar un error definir una clase abstracta sin método abstractos, pues esto permite que no se puedan generar instancias de esta clase durante el programa y a su vez que sus subclases no estén obligadas a redefinir los métodos.

d) Como *oa* es un arreglo de tipo ObjetoAstronomicoExtraSolar, se permite almacenar las instancias de sus subclases. En la línea 25, el método se invoca desde cada instancia almacenada en el arreglo, por tanto, ejecuta la versión especifica de cada instancia.

e) Debido a que oa[0] paso de estar referenciando a un objeto de tipo Galaxia a uno de tipo Supernova que esta referenciado por oa[2], y por tanto al ejecutar el método, este ejecuta el del objeto al que referencia.

f) La clase Estrella no necesita implementar descripción() debido a que esta también es abstracta, permitiendo que las subclases si tengan que definirlo.

g) Al definir el método tipoCuerpo1() como privado genera un error, debido a que los métodos abstractos deben ser protected o public, y las subclases que definan el método deben ser de igual o mayor visibilidad.

h) Nova hereda tipoCuerpo1() de Estrella, el cual puede sobrescribirlo pero al no necesitar un comportamiento en especifico no es necesario realizarlo.

i) Al llamar el método toString() en la línea 9 se obtiene un cadena de texto con la siguiente sintaxis: Galaxia@3af49f1c. Esto se debe a que todas las clases en Java heredan de Object, que define el método toString().

j) Se puede crear un puntero de tipo ObjetoAstronomicoExtraSolar debido a que este referencia una subclase concreta que en este caso es Nova. Pero no se puede instanciar una clase abstracta.

k) B: Invalida, debido a que no se puede crear una instancia de una clase abstracta.
C: valido, ya que se puede crear un puntero de una clase abstracta que referencia una subclase de la clase abstracta.

l) B: Correcta, pues se puede asignar un objeto de tipo Nova con un puntero de la clase abstracta por herencia.
C. Incorrecta, debido a que el método explotar no esta definido en la clase abstracta ObjetoAstronomicoExtraSolar.

Al omitir la instrucción C. se hace una especialización en el puntero, por lo que se puede utilizar el método explotar() imprimiendo así Boom!

m) Se imprime true debido a que todos los objetos son instancias de la clase Object. Sin embargo, en el código dado, al momento en el que obN referencia a null, la línea obN instanceof Object imprimiría false.
n) Si se pueden crear constructores en las clases abstractas, el cual se invoca mediante el super() en las subclases y se utilizan para inicializar los atributos que serán heredados por las subclases
o) Genera un error debido a que la clase EnanaBlanca no tiene el método abstracto explotar que le pertenece a su Padre. Para corregir el error, este método debe ser definido.