1. Analiza las siguientes asignaciones indicando si hay conversión explícita, implícita o ninguna:

```
double valor1;
int valor2;
long valor3;
float valor4;
char valor5;
 int valor6;
```

valor1 = 10 / 4; _____

valor2 = (int) (10.0 / 4); _____

valor3 = valor2 * 2L; _____

valor4 = valor3; _____

valor5 = 65; _____

valor6 = valor5 + 1; _____

2. Indica qué salida se mostrará por pantalla al finalizar la ejecución del siguiente código:

```
public static void main(String[] args) {
    int filas = 5;
    int i = 1;

    while (i <= filas) {
        String linea = "";
        int j = 1;
        while (j <= i) {
            if (i == filas) {
                linea += "* ";
            } else if (j == 1 || j == i) {
                linea += "* ";
            } else if (i % 2 == 0) {
                linea += "· ";
            } else {
                linea += "  ";
            }
            j++;
        }
        System.out.println(linea);
        i++;
    }
}
```

3. Completa el código siguiente para conseguir que esta pequeña aplicación funcione correctamente:

```
Import ____________________________;
import ____________________________;
import ____________________________;
import ____________________________;

public class EscribirArchivo {
   public static void main(String[] args) {

      String rutaFichero = System.getProperty("user.dir") + "/recursos/salida.txt";

      try (_______________bw = new _________________(new ___________________(___________________))) {
         String[] nombres = {"Ana", "Carlos", "María", "Pedro"};

         for (String nombre : nombres) {

            bw._____________ ("Alumno: " + nombre);

            bw.newLine();

         }
      } catch (_____________________________ e) {

         System.out.printf("El archivo no fue encontrado: %s", e.getMessage());

      } catch (_____________________________ e) {

         System.out.printf("Error de escritura: %s", e.getMessage());

      }

   }
}
```

4. Dada la siguientes implementaciones de clases e interfaces contesta a las cuestiones planteadas al final de la mismas:

Instrumento.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */

public abstract class Instrumento {
    protected String nombre;

    public Instrumento(String nombre) {
        this.nombre = nombre;
    }

     public String getNombre() {
        return this.nombre;
    }

    public abstract void tocar();

    // Método concreto (con implementación)
    public void afinar() {
        System.out.printf("Afinando el/la %s %n", this.getNombre());
    }

}
```

Conectable.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */
public interface Conectable {
    void conectar();
}
```

Percutible.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */
public interface Percutible {
    void golpear();
}
```

GuitarraElectrica.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */
public class GuitarraElectrica extends Instrumento implements Conectable {
    public GuitarraElectrica() {
        super("Guitarra Eléctrica");
    }

    @Override
    public void tocar() {
        System.out.println("🎸 Sonando riff distorsionado.");
    }

    @Override
    public void conectar() {
        System.out.println("Conectando al amplificador de 50W...\n");
    }
}
```

Bateria.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */
public class Bateria extends Instrumento implements Percutible {
    public Bateria() {
        super("Batería");
    }

    @Override
    public void tocar() {
        System.out.println("🥁 ¡Ba dum tss!");
    }

    @Override
    public void golpear() {
        System.out.println("Usando baquetas de madera de arce.\n");
    }
}
```

Piano.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */
public class Piano extends Instrumento implements Percutible {
    public Piano() {
        super("Piano de Cola");
    }

    @Override
    public void tocar() {
        System.out.println("🎹 Sonando melodía clásica.");
    }

    @Override
    public void golpear() {
        System.out.println("Las teclas activan los martillos internos.\n");
    }
}
```

Principal.java

```java
package instrumentosmusicales;

/**
 *
 * @author fcomo
 */
public class Principal {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Polimorfismo y Ligadura Dinámica
        Instrumento miGuitarra = new GuitarraElectrica();
        Instrumento miBateria = new Bateria();
        Instrumento miPiano = new Piano();

        GuitarraElectrica gEspecial = new GuitarraElectrica();


    }

}
```

1. Si intentamos ejecutar **`miBateria.golpear();`** directamente, ¿qué ocurrirá?.

2. Si eliminamos el método **`tocar()`** de la clase **`GuitarraElectrica`**, ¿qué sucedería?

3. Si implementamos la siguiente llamada **`miGuitarra.conectar();`** ¿sería correcta? ¿Por qué?

4. ¿Cómo se podría solucionar el problema del punto 1 para que **`miBateria`** pueda acceder al método golpear? (sin cambiar ningún diseño de clase o de interfaz).

5. ¿Qué ocurriría si intento llamar a miPiano.afinar()? ¿por qué?

6. Dibuja el esquema completo que represente correctamente todas las relaciones de la jerarquía de clases e interfaces proporcionada.