1. ¿Qué es mejor, almacenar la fecha de nacimiento o la edad? Razona tu respuesta

La fecha de nacimiento.

Porque es más sencillo crear un método que calcule la edad según su fecha de nacimiento que cambiar la edad todos los años.

2. ¿Cómo se implementan las operaciones de una clase?

Las operaciones se transformarán en constructores y métodos.

3. ¿Puede una clase tener varios constructores? ¿Para qué?

Sí, para poder instanciar objetos de distintas maneras, es decir, creando un objeto sin ninguna propiedad o con alguna a través de un mutador.

4. ¿Es correcta la expresión: ((b>4) * (c<2)? Razona tu respuesta

Sobra el primer paréntesis.

5. Escribe la signatura de 3 métodos que formen parte de la interface de la clase String. Consulta la documentación en la API de Java en Oracle.

```
public String getBytes(){
}
public String toString(){
}
public String hashCode(){
}
```

6. ¿Puede un accesor no recibir ningún parámetro? Pon un ejemplo y razona tu respuesta

```
public String getAlias() {
	return alias;
}
```

Sí, no puede recibir un parámetro, ya que solo nos interesa que nos muestre un valor de un atributo.

7. ¿Puede un mutador no recibir ningún parámetro? Pon un ejemplo y razona tu respuesta

```
private void setAlias(String alias) {
	assert alias !=null : “Error: El alias no puede ser nulo”;
	assert alias.lenght() > 0 : “Error: El alias no puede estar vacío”;
	this.alias = alias;
}
```

Debe recibir un parámetro para poder cambiar el valor del atributo.

8. ¿Qué pasaría si no cumpliéramos la ENCASULACION y OCULTACION de datos usando nivel de acceso public para los atributos de una clase?

Que los atributos podrían ser directamente modificables del exterior, con lo que la ocultación de datos se rompería y si ese atributo tiene una restricción, podría saltársela.

9. ¿Cuál es el ámbito de una variable de clase? ¿Y su tiempo de vida? Razona la respuesta

El ámbito de una variable de clase es toda la clase y su tiempo de vida es la ejecución del programa, ya que cuando se carga la clase en memoria, éstas se cargan también hasta que el proceso termine.

10. ¿Para qué serviría definir una variable local estática (static)? Haz una prueba y razona tu respuesta. Idea: comprueba si la variable tiene un tiempo de vida distinto al que esperas…

Porque nos permite conservar el valor de la variable hasta la próxima llamada de la función.

Java no permite crear variables locales estáticas ya que se restringen en el ámbito de clase.

11. ¿Puede un método de clase ser privado? Haz un ejemplo y razona tu respuesta.

Sí.

```
public class Alumno
{
	private String alias;
	private void setAlias(String alias) {
		assert alias !=null : “Error: El alias no puede ser nulo”;
		assert alias.lenght() > 0 : “Error: El alias no puede estar vacío”;
		this.alias = alias;
	}
}
```

Si queremos establecer un alias que nunca se cambie.

12. ¿Cómo se pueden controlar las restricciones sobre atributos?

Mediante aserciones en los mutadores.

13. Piensa e implementa un mutador que permita modificar un atributo de tipo String (supongamos que a nivel de diseño se ha establecido una restricción NOT NULL y otra NO VACIO sobre el atributo)

```
public class Alumno
{
	private String alias;
	private void setAlias(String alias) {
		assert alias !=null : “Error: El alias no puede ser nulo”;
		assert alias.lenght() > 0 : “Error: El alias no puede estar vacío”;
		this.alias = alias;
	}
}
```

14. ¿Es posible tener dos métodos en una clase cuyas signaturas sean iguales excepto el tipo de retorno? ¿Es un caso de sobrecarga?

No, porque no sabríamos a que método estamos llamando.

15. ¿Por qué el uso de aserciones no es la mejor solución para el control de restricciones en mutadores?

Porque para ello tenemos las excepciones.