1. La siguiente tabla propone un sumario de las constantes de clase de las **diferentes clases Wrapper** de tipo entero, a saber: **Byte**, **Short**, **Integer** y **Long**. A partir del objetivo requerido escribir el código que utilice el recurso oportuno.

#	Descripción	Código	Valor
1	Número de bytes de memoria que ocupa un dato de tipo primitivo **byte**		
2	Número de bytes de memoria que ocupa un dato de tipo primitivo **short**		
3	Número de bytes de memoria que ocupa un dato de tipo primitivo **int**		
4	Número de bytes de memoria que ocupa un dato de tipo primitivo **long**		
5	Máximo valor **positivo** que puede tener un dato de tipo primitivo **byte**		
6	Máximo valor **positivo** que puede tener un dato de tipo primitivo **short**		
7	Máximo valor **positivo** que puede tener un dato de tipo primitivo **int**		
8	Máximo valor **positivo** que puede tener un dato de tipo primitivo **long**		
9	Máximo valor **negativo** que puede tener un dato de tipo primitivo **byte**		
10	Máximo valor **negativo** que puede tener un dato de tipo primitivo **short**		
11	Máximo valor **negativo** que puede tener un dato de tipo primitivo **int**		
12	Máximo valor **negativo** que puede tener un dato de tipo primitivo **long**		
13	Instancia de la clase **Class** que representa el tipo primitivo **byte**		
14	Instancia de la clase **Class** que representa el tipo primitivo **short**		
15	Instancia de la clase **Class** que representa el tipo primitivo **int**		
16	Instancia de la clase **Class** que representa el tipo primitivo **long**		

2. La siguiente tabla propone una **lista de expresiones primitivas** que deben utilizarse con el proceso de **Boxing** para construir un objeto **Wrapper** de la clase **Integer**. Escribir el código correspondiente a cada caso.

#	Inicial	Código
1	`123`	
2	`43.21`	
3	`'W'`	
4	`true`	
5	`2 + 3`	
6	`20 / 3.0`	
7	`(char)('\u0041' + 4)`	
8	`!(2 <= 2)`	

3. La siguiente tabla propone una lista de objetos **Wrappers** de la clase **Integer** que deben utilizarse con el proceso de **UnBoxing** para obtener **el dato primitivo envuelto** correspondiente a cada caso.

#	Inicial	Tipo	Código	final
1	`Integer i = 123;`	`byte`		
2	`Integer i = 123;`	`short`		
3	`Integer i = 123;`	`int`		
4	`Integer i = 123;`	`long`		
5	`Integer i = 123;`	`float`		
6	`Integer i = 123;`	`double`		
7	`Integer i = 123;`	`char`		
8	`Integer i = 123;`	`boolean`		

4. La siguiente tabla propone una lista de objetos **Wrappers** de la clase **Integer** que deben utilizarse para generar un objeto de tipo **String** que sea su representación según los siguientes criterios.

#	Inicial	Criterio	Código	final
1	123	Base 2		
2	123	Base 8		
3	123	Base 10		
4	123	Base 16		
5	123	Base 5		
6	123	Unsigned		
7	-123	Unsigned		
8	123	Unsigned Base 16		
9	-123	Unsigned Base 16		

5. La siguiente tabla propone una lista de cadenas de texto con las que se pretende obtener el valor entero que representan en cada una de las bases correspondientes. Escribir el código necesario para realizar dicha conversión.

#	Inicial	Base	Código	final
1	"0b1010"	-		
2	"0753"	-		
3	"123"	-		
4	"0xFF"	-		
5	"PAKO"	-		
6	"0b1010"	2		
7	"0753"	2		
8	"123"	2		
9	"0xFF"	2		
10	"0b1010"	8		
11	"0753"	8		
12	"123"	8		
13	"0xFF"	8		
14	"0b1010"	16		
15	"0753"	16		
16	"123"	16		
17	"0xFF"	16		

6. La siguiente tabla propone una serie de operaciones que se pueden realizar con los recursos de la clase **Integer**. Escribir el código necesario para realizar dichas operaciones.

#	Inicial	Código	final
1	Relación entre 3 y 5		
2	Relación entre 5 y 3		
3	Relación entre 5 y 5		
4	Relación unsigned entre -3 y 5		
5	Relación unsigned entre 5 y -3		
6	Relación unsigned entre 5 y 5		
7	Máximo entre 3 y 5		
8	Máximo entre 5 y 3		
9	Máximo entre 5 y 5		
10	Mínimo entre 3 y 5		
11	Mínimo entre 5 y 3		
12	Mínimo entre 5 y 5		
13	Signo de 3		
14	Signo de -3		
15	Suma de 3 y 5		