Pregunta 1.

Los operadores que pueden ser usados con variables booleanas (*true/false*) son:

! , == Cast with (Boolean)

opción 4: ! Este operador invierte el valor de la variable, si es *false,* la convierte en *true* y viceversa.

opción 7: **Cast with (Boolean)** Es posible cast un valor String "true" / "false"a un boolean.

opción 1: **==** Es posible comparar dos objetos para rezar true o false si los objetos son iguales o apuntan al mismo espacio en la memoria, pero también se pueden comparar primitivos para ver si tienen el mismo valor, en este sentido sí se pueden aplicar a las variables booleanas.

opción 2 y 3: **+** y **-** no se pueden usar en una variable booleana porque el valor *true/false* no se puede incrementar, decrementar, sumar o restar.

opción 5: **%** módulo No puede aplicarse a boolean porque es el residuo de una división, no se puede aplicar una operación aritmética a un boolean.

opción 5: **<=** No se puede usar este operador con booleanos porque establece la relación del tipo mayor que o menor que y no hay mayor o menor que false o true.

Pregunta 2

Los data types que podrán funcionar con el código:

```
byte apples = 5;
short oranges = 10;

int / long  bananas = apples + oranges;
```

Boolean no es del tipo numérica así que no se puede resolver, *double* es numérica pero con punto decimal por lo que tampoco aplica. *Int y long* son las únicas *data types* que son mayores que short y byte por lo que el resultado de una suma de ambas cabe sin necesidad de cast explícito.

Pregunta 3

Cambios que permitirán compilar el código:

```
long ear = 10;
int hearing = 2 * ear;
```

Opción 2 hacer cast de ear to int → `int hearing = 2 * (int)ear; //` Long es más grande que int por eso se necesita castear como int.

Opción 3 cambiar el data type de ear a short → `short ear = 10; //` int es más grande que short por lo tanto cabe en un int

Opción 4 Hacer cast a 2 * ear a int → `int hearing = (int) (2 * ear);`

Opción 2 Cambiar el data type de hearing to long → `long hearing = 2 * ear; //` Siendo hearing long ya no se estará intentando meter un data type mas grande en uno más pequeño

Pregunta 4

Output:

5 No compila por la línea 9. Pero tampoco compila desde la fila 2 ya que el método dice que debe retornar un long, pero regresa un int.

Pregunta 5

Outputs:
4
5
1

Pregunta 6

Output:

El código no compila porque ticketSold es del tipo int y se le ha querido sumar un long que es mayor y no cabe.

De haberse ejecutado, ticketSold valdria 6 y ticketTaken 4.

Pregunta 7

Output:
Just Right

Pregunta 8

Which statement
break RABBIT
continue BUNNY
break

Pregunta 9

No compila por sintaxis, e¿keepGoing va entre paréntesis → while(keepGoing)

Pregunta 10

El código no termina. Por la linea 15 name.length() es mayor a 0 y se sugie aumentando así que nuca se detendrá.

Pregunta 11

El código no compila por la linea 5 esta declarada como String pero numFish es un int.

Pregunta 12

abbaccca.

Pregunta 13

En la linea 18 se manda una excepcion, ya que un String y un StringBuilder no se pueden comparar usando el operador ==

Pregunta 14

No compila, el método main no puede estar fuera de la clase.

Pregunta 14

puzzle.reverse(); (Aunque var es parte de una versión más adelantada de Java 8, en java 10; en el 8 no compila)