Oracle Certified Professional Java SE 17 Developer

Chapter 1: Building Blocks (pdf - 120)

1. D: public static final void main(String[] args)

   E: public static void main(String[] args)

   La opción E es la estándar pero se puede agregar el final aunque resulte redundante. Esta función tiene que ser pública, estática y retornar void

2. C: Z, Y, X

   D: Y, X

   E: Z, X

   El paquete y los imports son opcionales

3. A: Bunny es una clase

   E: bun es una referencia a un objeto

4. B: _helloWorld$

   E: Public

   G: _Q2_

5. A: El objeto creado en la línea 9 es elegible para garbage collection después de la línea 13

   D: El objeto creado en la línea 10 es elegible para garbage collection después de la línea 13

   F: El garbage collection puede funcionar o no

El objeto creado en la línea 10 no se elimina después de la línea 12 ya que sigue teniendo una referencia de brownBear.roar. El garbage collection no garantiza liberar la memoria

6. F: 7

   Todas las variables que se encuentran dentro de las mismas llaves: water, air, twoHumps, distance, path, age, i

7. C: La salida incluye “# cups = 0”

   Todo esta entre comillas, no se usan las variables

8. B: var fall = “leaves”;

   D: var night = Integer.valueOf(3);

   H: var morning = “”; morning = null;

   var acepta cualquier tipo de objeto pero no puede ser inicializado con null sin un tipo de dato. Var no puede ser usada para declarar varias variables en una sola línea

9. E: Una variable de clase String por default es null

   Las variables locales no tienen valores por default. El valor por default de un float es 0.0. El valor por default de los tipos primitivos nunca es null

10. A: 3_1, E: 2_234.0_0, F: 9___6

    Los _ se usan para separar números pero no pueden ir al inicio, al final, antes ni después de un punto

11. E: 4

12. A: La línea 2 genera un error de compilador

    C: La línea 4 genera un error de compilador

    D: La línea 7 genera un error de compilador

    Para declarar dos tipos de datos diferentes se debe hacer separándolos con ;. No se puede declarar el valor de un parámetro en la firma de la función. No se puede imprimir una variable a la que no se tiene acceso

13. A: import Aquarium.*;

    C: import Aquarium.*; import aquarium.jellies.Water;

    D: import Aquarium.*; import aquarium.jellies.*;

14. A: La línea 3 genera un error en el compilador

    B: La línea 4 genera un error en el compilador

    D: La línea 6 genera un error en el compilador

    E: La línea 7 genera un error en el compilador

    No se puede convertir un long en short sin casteo explícito. No se puede convertir un decimal en int sin casteo explícito. short no tiene un método length(). int no tiene un método length

15. C: El garbage collection permite al JVM reclamar memoria para otros objetos

    E: Un objeto puede ser elegible para el garbage collection y nunca ser removido de la memoria

F: Un objeto puede ser elegible para garbage collection una vez que no tenga referencias accesibles a él en el programa

16. A: Imprime 2 líneas

D: Hay una línea con un espacio blanco al final

Solo tiene un salto e línea (\s). Hay un espacio antes del \s

17. D: Empty = false, F: Brand = null, G: Code = 0.0

Son los valores por default

18. B: El tipo de var es conocido en tiempo de compilación

C: No se puede usar como una variable de instancia

F: Su tipo no puede cambiar en tiempo de ejecución

G: var es una palabra reservada en java

19. A: La salida es 100

D: num1 es un número primitivo

Compara dos números iguales. Long.parseLong() convierte en long (primitivo), mientras que Long.valueOf()convierte en Long (objeto)

20. C: P1 imprime null

A pesar de que no lo marca como un error, el constructor no devuelve ningún valor, por lo que no se inicializan las

variables de los objetos, entonces solo existen los valores por default (0 y null)

21. D: 7-0-2-4-

Al iniciar el programa se imprime la línea 10 (7-), después se crea un objeto que hace que se imprima la línea 3 (0-) con un valor por default porque la variable aún no pasa por el constructor. Luego pasa por el constructor, donde se imprime la línea 7 (2-). Finalmente, continuando con la función main, imprime la línea 12 (4-), el contador que se inicializo cuando pasó por el constructor

22. C: int amount = 0xE;

F: int amount = 0b101;

G: double amount = 9_2.1_2;

La opción A no declara la variable que se usa más adelante. Las opciones B y D requieren casteo implícito. Las opciones E y H tienen una sintaxis incorrecta

23. A: La línea 3 genera un error

D: La línea 10 genera un error

Le falta la f al float. La variable se declaro dentro de un bucle, es inalcanzable fuera de él

Chapter 2: Operators (pdf - 160)

1. A: ==, D: !, G: Casteo con (boolean)

2. A: int, B: long, D: double

   La suma de dos números no da un valor booleano, da un valor numérico. Short y byte son tipos de datos muy pequeños

3. B: Casteo implícito (int) a ear,

   C: Cambiar el tipo de dato de ear a short

   D: Casteo implícito (int) a (2 * ear)

   F: Cambiar el tipo de dato de hearing a long

   No compilaría sin modificar nada ya que convertir un long en int requiere un casteo explícito. Cambiar el tipo de dato de hearing a short no compilaría ya que convertir un long en short requiere casteo explícito

4. B: true, 20, false

   canine = (teeth != 10) ∧ (wolf = false)

   ```
   //canine = (true) ∧(false)
   //canine = true
   ```

5. A: +, *, %, --

   C: =, ==, !

6. F: Ninguna de las anteriores

   El programa no compila porque el resultado en la línea 3 termina en float gracias al casteo que convierte el double

en int. Para convertir un float en long (lo que devuelve la función) se requiere un casteo explícito

7. D: true, false, false

```
int ph = 7, vis = 2;
boolean clear = vis > 1 & (vis < 9 || ph < 2);
              // true & (true || false)
               // true & (true)
               // true
boolean safe = (vis > 2) && (ph++ > 1);
             // (false) && (true)
              // false
              // ph no se modifica
Boolean tasty = 7 <= --ph;
             // false
              // ph = 6
```

8. A: 4 - 1

```
pig = pig++; //El post incremento no altera el valor de la variable
goat -= 1.0; //Esta forma sí altera el valor de la variable
```

9. A: 1, D: 4, E: 5

```
int a=2, b=4, c=2;
System.out.println(a>2 ? --c : b++);
                 //(false -> b++) = 4
                 //Primero hace la impresión y luego el incremento
                 //No se modifica b
System.out.println(b = (a!=c ? a : b++));
                  //(b = (false -> b++))
                  //(b = b++)
                  //Primero se resuelve la operación y se imprime
                  //b=5
System.out.println(a>b ? b<c ? b : 2 : 1);
                  //false -> 1
```

10. G: El código no compila

No se pueden hacer operaciones entre bytes debido a la posible pérdida de datos

11. D: 2, 0, 5

```
int sample1 = (2 * 4) % 3; // = (8) % 3 = 2
int sample2 = 3 * 2 % 3; // = 6 % 3 = 0
int sample3 = 5 * (1 % 2); // = 5 * (1) = 5
```

12. D: post-incremento, pre-decremento

El post-incremento aumenta el valor y devuelve el valor original, mientras que el pre-decremento disminuye un valor y devuelve el nuevo valor

13. F: true – true – false

```
boolean sunny = true, raining = false, sunday = true;
boolean goingToTheStore = sunny & raining ^ sunday;
                          //true & false ^ sunday
                          //false ^ true
                          //true
boolean goingToTheZoo = sunday && !raining;
                        //true && true
                        //true
boolean stayingHome = !(goingToTheStore && goingToTheZoo);
                      //!(true && true)
                      //!(true)
                      //false
```

14. B: != se puede usar para comparar objetos

D: En tiempo de ejecución, & y | pueden hacer que solo el lado izquierdo de la expresión se evalue

E: El valor de retorno de una operación de asignación es el valor de la nueva variable asignada

G: ! no puede invertir valores números

15. D: ? :

16. B: 1

Se necesita un casteo implícito para pasar de long a int

17. C: ticketsSold es 6

F: ticketsTaken es 4

```
int ticketsTaken = 1;
int ticketsSold = 3;
ticketsSold += 1 + ticketsTaken++; // = 5
//Primero se hace la operación y luego se incrementa
//ticketsTaken = 2
ticketsTaken *= 2; // = 4
ticketsSold += (long)1; // = 6
```

18. A: []

19. B: start es -128, F: end es 12

```
int start = 7;
int end = 4;
end += ++start; //end += 8 //end = 12
start = (byte)(Byte.MAX_VALUE + 1); //= (byte)(128) = -128
```

20. A: Se ejecutan antes de los operadores binarios y ternarios numéricos

D: El post-decremento devuelve el valor de la variable antes de aplicar el decremento

E: ! no se puede usar en valores numéricos

21. E: -9, -8, 9

```
int myFavouriteNumber = 8;
int bird = ~myFavouriteNumber; //= ~1000 = -1001 = -9
int plane = -myFavouriteNumber; //= -8
var superman = bird == plane ? 5 : 10; //= false -> 10
System.out.println(bird+”,”+plane+”,”+ --superman);
                                          //superman = 9
```

Chapter 3: Making Decisions (pdf - 208)

1. A: enum, B: int, C: byte, E: String, F: char, G: var

2. B: Just Right

```
int temperature = 4;
long humidity = -temperature + temperature * 3;
            // = -4 + 12 = 8
if(temperature >= 4) //true
if(humidity < 6) System.out.println("Too low"); //false
else System.out.println("Just right"); //esto es lo que imprime
else System.out.println("Too high");
```

3. A: Double[][], D: List, F: char[], H: set

   for-each recorre colecciones de datos, aunque no todas

4. F: Ninguna de las anteriores

   Se requiere un default dentro del switch

5. E: Exactamente una línea del código no compila

   El código que esta después del continue es inalcanzable ya que ahí se sale del bucle

6. C: La expresión condicional de un ciclo for es evaluada antes de la primera ejecución

   D: Si un switch expression usa una cadena y asigna el resultado a una variable, requiere la opción default

   E: do-while garantiza que el código se ejecuta al menos una vez

7. B: int i=0; i<=weather.length-1; ++i

   D: int i=weather.length-1; i>=0; i--

   La primera opción tendría que restar 1 en la inicialización del entero y hacer que siga mientras sea mayor o igual a 0.

   La tercera opción daría error porque nunca se declara i.

   La quinta opción usa dos veces la palabra int.

   La sexta opción solo imprime los elementos del 1 al 9, sin importar el tamaño del arreglo, a menos que el arreglo sea más chico

8. G: El Código tiene dos líneas que no compilan

   La variable bat en la línea 36 no existe después del || ya que aún no se confirma su existencia. Usa default en lugar de else

9. B: break RABBIT

   C: continue BUNNY

   E: break

   La primera opción imprimiría 1. La cuarta opción imprimiría 5. La sexta opción imprimiría 5.

10. E: 4

    No se puede usar continue dentro de switch. Si se usa una constante (thursday) tiene que ser conocida por el compilador. Se conoce el valor de Sunday pero no es una constante. DaysOfWeek.MONDAY no es un entero

11. A: 3

El código se ejecuta sin errores

12. C: 23

```
int sing = 8, squawk = 2, notes = 0;
while(sing > squawk)
{
     sing--;
     squawk += 2;
     notes += sing + squawk;
}
```

True	True	False
7	6	
4	6	
11	23	

13. G: El Código no compila por una razón diferente

La condicional después de while debe estar entre paréntesis

14. B: El tipo de dato de penguin es int

D: El tipo de dato de emu es Character

F: El tipo de dato de macaw es Integer

```
for(var penguin : new int[2])
     System.out.println(penguin); //0 0
var ostrich = new Character[3];
for(var emu : ostrich)
     System.out.println(emu); //null null null
List<Integer> parrots = new ArrayList<Integer>();
for(var macaw : parrots)
     System.out.println(macaw); //no imprime nada
```

15. F: Ninguna de las anteriores

El código no compila poque se requiere un case antes de ‘C’

16. A, B, D

Las opciones 3 y 6 compilan pero se salen de los límites del arreglo. La opción 5 crea un bucle infinito, ya que mantiene la variable del for con el mismo valor con el que la inicializa

17. B: 3, E: 10

```
int participants = 4, animals = 2, performers = -1;
while((participants = participants + 1) < 10) {}
//participants = 5 -> 6 -> 7 -> 8 -> 9 -> 10
do{} while(animals++ <= 1);
//animals = 3
for(; performers < 2; performers += 2) {}
//performers = 1 -> 3
```

18. C: La coincidencia de patrones con if es implementada usando instanceOf

E: El alcance del flujo significa que una variable de patrón solo es accesible si el compilador puede definir su tipo

19. E: El Código no compila

La variable que evalúa while no está a su alcance

20. A: break L2 en la línea 8; continue L2 en la línea 12

E: continue L2 en la línea 8; continue L2 en la línea 12

Las opciones B y D provocan bucles infinitos. La opción C causa un error ya que llama al label L3 antes de que se declare

21. D: tres

En la línea 23 hay que quitar las llaves y agregar ; al final. En la línea 24 hay que quitar las llaves y un ; del final, además, la variable yiel no está declarada. El case 30 está duplicado en las líneas 25 y 26

22. E: 5 2 1

Al entrar en el switch, se imprime “5 ”, después pasa por el while con un valor de 3. Se imprime “2 ”. Vuelve a entrar por el while con un valor de 2. Se imprime “1 ” y ya no puede volver a entrar al while

23. F: ninguna de las anteriores

El código no compila porque la sentencia else if se encuentra después de else

24. G: Ninguna de las anteriores

El for no está declarado correctamente

25. D: 2

Cuando entra en el switch y encuentra el valor que corresponde, no se detiene hasta que acaba o encuentra un break;

26. F: El código compila pero nunca termina

El ciclo do-while se vuelve infinito ya que la variable r nunca cambia dentro del bucle

27. F: El código no compila

Los case no deberían ir entre llaves

28. F: Ninguna de las anteriores

El código no compila porque se esta creando la misma variable dos veces, en las líneas 41 y 43

29. C: -1 0 1 2 3 4 5 6