DEPURACIÓN DE PROGRAMAS JAVA

Despois de codificar un programa Java, hai que comprobar que compila e funciona correctamente. Un **bug** no programa é un erro no programa. A depuración é o proceso de atopar e solucionar estes erros.

Tipos de erros nos programas

Un **erro do compilación** prodúcese cando o compilador non entende unha sentenza do programa. Estes erros son a miúdo o resultado de escribir erroneamente palabras, espazamentos incorrectos, uso de maiúsculas ou minúsculas incorrectamente ou falta de signos de puntuación. A mellor forma de minimizar estes erros é prestar moita atención aos detalles ao escribir os programas.

Exemplos	
`doubel height;`	Palabra clave **double** mal escrita
`double height`	Falta o punto e coma (;)
`Double height;`	Uso de maiúsculas cando deben usarse minúsculas
`public class Ola Mundo`	Uso incorrecto do espazamento

Un **erro en tempo de execución** (run-time error) non o detecta o compilador pero fai que o ordenador pare de forma anormal a execución do programa.

Exemplo

Indicar un número que non sexa un número en punto flotante fai que o seguinte código falle.

```
java.util.Scanner in = new java.util.Scanner( System.in);
System.out.print("Indica o radio do circulo: ");
double radius = in.nextDouble( );
```

Un **erro lóxico** é un erro na lóxica do programa que fai que a saída do mesmo sexa incorrecta.

Exemplo

A seguinte sentenza calcula incorrectamente a velocidade en km/h. O erro non é obvio pero a saída do programa será incorrecta.

```
kph = horas/kms;
```

Depurar erros de compilación

Cando o compilador atopa un erro nun programa xera unha **mensaxe de diagnóstico**. O diagnóstico é unha pista que indica que hai un erro pero algunhas veces non é moi preciso.

Un diagnóstico pode citar un número de liña incorrecto, dar unha explicación pouco clara ou incluso totalmente errónea.

Exemplo

Típico diagnóstico Java:

```
KPH1.java:10: cannot find symbol
symbol    : variable Kms
location: class KPH1
        Kms = 341;
        ^
```

Na primeira liña temos o nome do ficheiro Java, o número de liña da sentenza que da erro e a descrición do erro.

Na segunda liña temos o símbolo descoñecido e o tipo de símbolo.

A terceira liña indica a clase na que aparece o erro.

A cuarta liña reproduce a sentenza Java que produce o erro e a quinta liña coloca unha marca (^) debaixo do símbolo que produce o erro.

O compilador non xera o ficheiro executable con extensión class a menos que o ficheiro fonte Java non produza erros de compilación.

Unha compilación sen erros é unha compilación limpa.

Recoméndase corrixir o primeiro erro que da o compilador e recompilar se os restantes erros non teñen sentido. Isto débese a que os diagnósticos restantes poden ser espurios – un diagnóstico espurio pode dar por incorrecta unha sentenza correcta.

Unha avalancha de diagnósticos espurios dise que é unha cascada de erros.

Polo tanto, debemos concentrarnos no primeiro diagnóstico. É o único que podemos considerar auténtico.

Exemplo

A seguinte aplicación ten un erro (falta unha chave ({) que se borrou na liña 4. Este único erro produce 28 mensaxes de diagnóstico do compilador co JDK 1.8.

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

     // declarar variables
      double kms;      // kms recorridos nun coche
      double     horas;//numero de horas que levou recorrer os kms
      double     kph;  //km/h
      // inicializar datos
     kms = 341;
     horas = 15.5;
     // calcular kph
      kph = horas / kms;
      // visualizar resultados
       System.out.print( kms + " km." );
       System.out.print( horas + " h" );
       System.out.println( " = " + kph + " km/h" );
    }
}
```

Depurar erros en tempo de execución

Se durante a execución do programa se produce unha excepción, é dicir, unha situación inesperada que non foi programada, detense a execución e xérase unha mensaxe de erro.

Exemplo

A seguinte mensaxe Java en tempo de execución obtense cando se utiliza un tipo de dato erróneo.

```
    Exception in thread "main" java.util.InputMismatchException
            at java.util.Scanner.throwFor(Scanner.java:909)
            at java.util.Scanner.next(Scanner.java:1530)
            at java.util.Scanner.nextDouble(Scanner.java:2456)
            at KPH.main(KPH.java:15)
```

As liñas informan exactamente de onde se produciu a excepción e como se propagou a través da cadea de chamadas a métodos.

Neste caso, o método que fallou foi KPH.main do ficheiro KPH.java na liña 15.

Depurar erros lóxicos

Un programa que se executa correctamente de principio a fin dise que remata de maneira normal. Non obstante, o programa pode non ser correcto se ten un erro lóxico. Os erros lóxicos non producen mensaxes de erro. A única maneira de detectalos é revisar as saídas que produce o programa.

Exemplo
Imaxinemos un programa que imprima: 15.5 horas/341.0 kms=22.0 kph Como 15.5/341.0=0.045 ≠ 22, a saída é claramente errónea.

Para atopar estes erros compárase a saída do programa co resultado que se supón que debe saír.

Sempre debemos usar datos de entrada para os que sabemos a resposta correcta para poder comprobar o funcionamento do programa.

Para depurar os erros lóxicos podemos usar o depurador para percorrer o programa paso a paso e ver o contido da memoria en cada paso. Podemos comparar os nosos datos calculados previamente cos que devolve o depurador para atopar onde falla o programa.

Exercicios

1. Compila a aplicación KPHx. Observa a cascada de erros. Aparece un número de liña correcto no primeiro diagnóstico?
2. Compila a aplicación **AnalizarDiagnostico** que aparece a continuación. Cal é o diagnóstico? Que liña aparece no mesmo? É fidedigna a explicación?

```
public class AnalizarDiagnostico
{
public static void main( String [] args )
    {
      double coste, cant;
      java.util.Scanner in = new java.util.Scanner( System.in );
         System.out.print("Indica unha cantidade: ");
         cant = in.nextDouble( );
         coste = 15,000€;
         System.out.println( "Total: " + cant * coste);
    }
}
```

3. Corrixe a aplicación **AnalizarDiagnostico** do exercicio anterior. Compila e executa a aplicación. Realiza posteriormente os seguintes cambios:
   a) Na liña 5 cambia **double** a **doubel**. Recompila a aplicación e observa o diagnóstico. Desfai o último cambio feito.
   b) Na liña 8 cambia **cant** a **Cant**. Recompila a aplicación e observa o diagnóstico. Que significa o diagnóstico **cannot find symbol**? Desfai o último cambio feito.
   c) Na liña 1 cambia **AnalizarDiagnostico** por **Analizar Diagnostico**. Recompila a aplicación e observa o diagnóstico. Desfai o último cambio feito.
4. Practica depurando o seguinte programa. Ten aproximadamente 10 erros de compilación. Identifícaos e corríxeos

```
import java.util.Scanner;
public class Multiplicadorx
{
    public static void main( String [] args )
    {
        // declara datos
        double num1; double num2; double produto;
        // indica datos
        java.util.Scanner in = new Scanner( System.in );
        System.out.print("Indica o primeiro número a multiplicar:
");
        double num1 = in.nextdouble( );
        System.out.print( "Indica o segundo número a multiplicar:
" );
        double num2 = in.nextdouble( );
        // calcula resultado
        produto = num2º * num1º;
        // mostra resultados
        System.out.print( Num1 + " * " );
        System.out.print( Num2 + " = " );
        System.out.println( produto );
    }
}
```

5. Practica depurando o seguinte programa. Ten uns 22 erros de compilación, un erro lóxico e un erro de usabilidade. Identifica e corrixe todos os erros.

```
import java.util.scanner;
This program finds the average of three input values.
public class Aritmeticax
{
    public static void main( String [] args )
        //declaración de variables
        double num1, num2, num3;
        //entrada de datos
        scanner in = newscanner( );
        System.out.print( "Number 1? " );
        System.out.print( "Number 2? " );
        System.out.print( "Number 3? " );
        number 1 = in.nextDouble;
        number 2 = in.nextDouble;
        number 3 = in.nextDouble;
        //calcula a media
        media = num1 + num2 + num3 / 3;
        //imprime o resultado
        system.out.print( Num1 + ", " + Num2 + " & " + Num3 );
        system.out.println( " ten como media " + media );
}
```

6. Practica depurando o seguinte programa. Ten aproximadamente 8 erros de compilación, 3 erros lóxicos, 3 erros de usabilidade e 1 erro relacionado coa lectura do programa. Identifica e corrixe todos os erros.

```
import java.util.Scanner;
//    Convirte unha temperatura de Fahrenheit a Celsius.
public class FahrenheitACelsiusx {
    public static void main( String [] args ) {
        // declaración de variables
        double celsius; double fahrenheit;
        // entrada de temperatura
        Scanner in = new Scanner( System.in );
        double celsius = in.nextDouble( );
        // calcula o equivalente celsius
        double celsius = 5/9 * fahrenheit - 32;
        // mostra os resultados
        System.out.println( fahrenheit, "\U00B0F = " );
        System.out.println( celcius, "u00B0C" );
    }
}
```

7. O seguinte programa ErroLoxico ten un erro lóxico. Busca o erro e corríxeo. Por que se produce o erro lóxico?

```
public class ErroLoxico {
    public static void main(String args[]) {
      int num1=10;
      int num2=25;
      System.out.println("A suma e " +num1+num2);
    }
}
```

8. Observa o seguinte programa Maior3. Funciona correctamente? Fai probas con varios casos de proba. No caso de que teña un erro lóxico, corríxeo.

```
import java.util.Scanner;
public class Maior3 {
    public static void main(String args[]) {
        Scanner in=new Scanner(System.in);
        int num1,num2,num3,max=0;
        System.out.println("Introduce un numero:");
        num1=in.nextInt();
        System.out.println("Introduce un segundo numero:");
        num2=in.nextInt();
        System.out.println("Introduce un terceiro numero:");
        num3=in.nextInt();
        if ((num1>num2) && (num1>num3)){
            max=num1;
        }else if ((num2>num1) && (num2>num3)){
            max=num2;
        }else{
            max=num3;
        }
        System.out.println("O maior numero e " +max);
    }
}
```

9. O seguinte código ten un erro de compilación. Podes atopalo?

```
public class ExemploMalo(){
       public static void main(String[] args){
              System.out.println("Codificando!");
       }
}
```

10. Observa o seguinte programa. Cres que funciona correctamente? Fai probas con varios casos de proba. No caso de que teña un erro lóxico, corríxeo.

```
import java.util.Scanner;
public class SumaPar {
    public static void main(String args[]) {
        Scanner in=new Scanner(System.in);
        int num1,num2;
        boolean ePar;
        System.out.println("Introduce un numero:");
        num1=in.nextInt();
        System.out.println("Introduce un segundo numero:");
        num2=in.nextInt();
        if ((num1+num2)%2==2){
            ePar=false;
        }else{
            ePar=true;
        }
        System.out.println("Suma par:" +ePar);
    }
}
```

11. Corrixe os erros que poda ter o seguinte programa.

```
public class ContadorArray{
     public static void main(String[] args){
          int num1 = 1000000000;
          int[] array = {num1, num1, num1};
          int total=0;
          for(int i = 1; i < array.length; i++){
               total += array[i];
          }
          System.out.println("O total e: " + total);
     }
}
```

12. Que tipo de erro se produce no seguinte programa? Como podemos resolvelo?

```
public class Media{
    public static void main(String[] args){
        int cont = 0, suma = 100, media;
        media = suma / cont;
    }
}
```

13. É correcto o seguinte código? En caso de que non sexa correcto, corríxeo.

```
public class Condicion{
    public static void main(String[] args){
        boolean condicion = true ;
        int cont=0;
        while (condicion = true)
        {
            System.out.println("si");
            cont++;
            if (cont==10){
                condicion=!condicion;
            }
        }
    }
}
```

14. Observa o seguinte código. Como solucionarías o problema de desbordamento?

```
public class DesbordamentoTest {
   public static void main(String[] args) {
      // O rango dos int e [-2147483648, 2147483647]
      int i1 = 2147483647;        // int maximo
      System.out.println(i1 + 1);  // -2147483648 (desbordamento)
      System.out.println(i1 + 2);  // -2147483647 (desbordamento)
      System.out.println(i1 + 3);  // -2147483646 (desbordamento)
      System.out.println(i1 * 2);  // -2 (desbordamento)
      System.out.println(i1 * i1);  // 1 (desbordamento)

      int i2 = -2147483648;         // int maximo
      System.out.println(i2 - 1);   // 2147483647 (desbordamento)
      System.out.println(i2 - 2);   // 2147483646 (desbordamento)
      System.out.println(i2 * i2);  // 0 (desbordamento)
   }
}
```

15. Cal é o problema do seguinte código?

```
public class ProblemaArray{
    public static void main(String[] args){
        int[] arr = {3,4,5} ;
        System.out.println( arr[3] ) ;
    }
}
```

16. Por que falla o seguinte programa? Como se resolve o problema?

```
public class Promedio {
    public static void main(String[] args) {
        int a = 3;
        int b = 4;
        int c = 20;
        int media;
        media = (a + b + c)/5.0;
        System.out.println(media);
    }
}
```

17. Resolve os problemas que se producen en tempo de execución co seguinte código.

```
public class Proba {
    System.out.println("Ola!");
    public static void main(String[] args) {
        System.out.println("Mundo!");
    }
}
```

18. Cal é o problema do seguinte código? Resólveo.

```
public class Proba2 {
    public static void main(String[] args) {
        int x = 2;
        int y;
        System.out.println(x + y);
    }
}
```

19. Corrixe os erros que teña o seguinte programa.

```
public class Test {
    public static void main(String[] args) {
        int x;
        boolean setX = false;
        if (setX) {
            x = 10;
        }
      System.out.println(x);
    }
}
```

20. O seguinte programa que serve para calcular os factores primos dun número enteiro grande ten varios tipos de erros. Arránxao.

    Este tipo de factorización úsase moito en criptografía.

    Exemplos de saídas correctas para o programa serían:

    3.757.208=2 x 2 x 2 x 7 x13 x 13 x 397

    98= 2 x 7 x 7

    17 = 17

```
public class Factores {
    public static void main(String args[]) {
        int n=Integer.parseInt(args[0]);
        for (i=0; i<n;i++)
        {
            while (n%i==0)
                System.out.print(i+" ")
                n=n/i
        }
    }
}
```