EJERCICIO 1

a) ¿Una clase puede heredar de dos clases en Java?

b) ¿Una interfaz Java puede tener métodos que incluyan una sentencia while? ¿Una interfaz Java puede tener métodos que incluyan una sentencia System.out.println?

c) ¿Un objeto Java puede ser del tipo definido por una interfaz? ¿Un objeto Java puede ser al mismo tiempo del tipo definido por una interfaz y del tipo definido por una clase que no implementa la interfaz? ¿Un objeto Java puede ser al mismo tiempo del tipo definido por una interfaz y del tipo definido por una clase que implementa la interfaz?

EJERCICIO 2

Construir una clase *ArrayReales* que declare un atributo de tipo double[] y que implemente una interfaz llamada *EstadisticasIF*. El contenido de esta interfaz es:

```
public interface EstadisticasIF {
   double minimo();
   double maximo();
   double sumatorio(); }
```

EJERCICIO 3

Construir una clase *Math3* que implemente una interfaz llamada *ExtremosIF* tal que:

```
public interface ExtremosIF {
   int min(int[] a); // devuelve el entero mínimo del array de enteros
   int max(int[] a); // devuelve el entero máximo del array de enteros
   double min(double[] a); // devuelve el real mínimo del array de reales
   double max(double[] a); } // devuelve el real máximo del array de reales
```

EJERCICIO 4

Construir una interfaz *RelacionesIF*, y después, una clase que la implemente llamada *Relaciones*, que incluya los siguientes métodos:

```
// devuelve verdadero si A es mayor que B
boolean esMayor(Object a, Object b) ;
// devuelve verdadero si A es menor que B
boolean esMenor(Object a, Object b) ;
// devuelve verdadero si A es igual que B
boolean esIgual(Object a, Object b) ;
```

SOLUCIÓN EJERCICIO 1

a) No. Java no permite la herencia múltiple de clases, por lo que una clase puede heredar máximo de 1. Sin embargo, con las interfaces se puede simular la herencia múltiple, puesto que permiten que una clase implemente varias interfaces.

b) Una interfaz, por defecto, solo puede tener métodos abstractos, es decir, métodos sin cuerpo. Dado que las sentencias siempre van en el cuerpo de los métodos, no sería posible. Sin embargo, con Java 8 y los métodos *default*, ahora es posible tener métodos no abstractos, y por tanto, cuerpo para los metodos declarados *default*.

c) Si, un objeto (o variable) puede ser del tipo de una interfaz determinada. Debido a la simulación de herencia múltiple comentada arriba, un objeto puede ser del tipo de una interfaz, y a su vez del tipo de otra clase de la que herede, implemente esta última clase la interfaz o no la implemente.

SOLUCIÓN EJERCICIO 2

Código fuente para la solución del ejercicio 2:

```
public class ArrayReales implements EstadisticasIF {
   double[] valor;

   public void asignar() {
      for (int i = 0; i < valor.length; i++) {
         valor[i] = Math.random();
      }
   }

   public double minimo() {
      double menor = valor[0];
      for (int i = 0; i < valor.length; i++) {
         if (menor > valor[i]) {
            menor = valor[i]; } }
      return menor; }

   public double maximo() {
      double mayor = valor[0];
      for (int i = 0; i < valor.length; i++) {
         if (mayor < valor[i]) {
            mayor = valor[i]; } }
      return mayor; }

   public double sumatorio() {
      double suma = 0.0;
      for (int i = 0; i < valor.length; i++) {
         suma += valor[i]; }
      return suma; }

   public void imprimir() {
      for (int i = 0; i < valor.length; i++) {
         System.out.println("x[" + i + "]=" + valor[i]); } } }
```

SOLUCIÓN EJERCICIO 3

Código fuente para la solución del ejercicio 3:

```
public final class Math3 implements ExtremosIF {
   public int min(int[] a) {
      int menor = a[0];
      for (int i = 1; i < a.length; i++) {
         if (menor > a[i]) {
            menor = a[i];  } }
      return menor; }

   public int max(int[] a) {
      int mayor = a[0];
      for (int i = 1; i < a.length; i++) {
         if (mayor < a[i]) {
            mayor = a[i];
         } }
      return mayor; }

   public double min(double[] a) {
      double menor = a[0];
      for (int i = 1; i < a.length; i++) {
         if (menor > a[i]) {
            menor = a[i]; }  }
      return menor; }

   public double max(double[] a) {
      double mayor = a[0];
      for (int i = 1; i < a.length; i++) {
         if (mayor < a[i]) {
            mayor = a[i]; }  }
      return mayor; } }
```

SOLUCIÓN EJERCICIO 4

Código fuente solución al ejercicio 4:

```
public interface RelacionesIF {

   boolean esMayor(Object a, Object b) ; // devuelve verdadero si A es mayor que B

   boolean esMenor(Object a, Object b) ; // devuelve verdadero si A es menor que B

   boolean esIgual(Object a, Object b) ; } // devuelve verdadero si A es igual que B


// suponemos que estamos comparando Integer

public class Relaciones implements RelacionesIF {

   boolean esMayor(Object a, Object b) {

      return (((Integer) a).intValue()  - ((Integer) b).intValue()) > 0; }


   boolean esMenor(Object a, Object b) {

      return (((Integer) a).intValue()  - ((Integer) b).intValue()) < 0; }


   boolean esIgual(Object a, Object b) {

      return (((Integer) a).intValue()  - ((Integer) b).intValue()) == 0;  } }
```