Ejercicio U8_B6_E1:

```
import java.util.ArrayList;
import java.util.List;

class App{
   public static void main(String[] args){
      List<String> items = new ArrayList<>();
          items.add("A");
          items.add("B");
          items.add("C");
          items.add("D");
          items.add("E");
      items.forEach(item->System.out.println(item));
   }
}
```

Observa que con el for mejorado tradicional hago la declaración String item. En el foreach el tipo de item se infiere: ya que items es una List<String> el tipo de item es String

Ejercicio U8_B6_E2:

Como cada item es un String, y como la clase String ya tiene sobreescrito el método toString(), en este caso, obtener el resultado requerido es inmediato

```
import java.util.ArrayList;
import java.util.List;

class App{
   public static void main(String[] args){
      List<String> items = new ArrayList<>();
          items.add("A");
          items.add("B");
          items.add("C");
          items.add("D");
          items.add("E");
      items.forEach(System.out::println);
   }
}
```

Ejercicio U8_B6_E3:

```
import java.util.ArrayList;
import java.util.List;
import java.util.function.Consumer;

class App{
   public static void main(String[] args){
      List<String> items = new ArrayList<>();
          items.add("A");
          items.add("B");
          items.add("C");
          items.add("D");
          items.add("E");
      //ojo: al contener un if necesito usar {}
      //ojo: si el cuerpo lambda lleva {} hay que acabar con ;
          Consumer<String> consumer=x->{if(x.equals("B"))
                        System.out.println(x);};
      for(String item : items){
         consumer.accept(item);
      }
      //dos formas equivalentes con foreach
      items.forEach(consumer);
      items.forEach(x->{if(x.equals("B"))
                        System.out.println(x);});

   }
}
```

Aunque es posible, normalmente las expresiones lambda no contienen if, ni mucho menos bucles. El ejemplo anterior se hace mejor utilizando streams, donde el if va a conseguirse realmente con una operación de filtrado... ¡Lo veremos en un boletín posterior!

Ejercicio U8_B6_e4:

¡qué fácil recorrer un mapa! ¡adiós a entrySet()!

```
import java.util.HashMap;
import java.util.Map;

class App{
   public static void main(String[] args){
      Map<String, Integer> items = new HashMap<>();
          items.put("A", 10);
          items.put("B", 20);
          items.put("C", 30);
          items.put("D", 40);
          items.put("E", 50);
          items.put("F", 60);

          items.forEach((k,v)->System.out.println("clave "+k+" con valor"+v));
  }
}
```

En este caso el Map es un HashMap, pues bien, la clase HashMap() tiene que tener una implementación del método forEach(probablemente sea a base de entrySet()), pero a nosotros lo único que nos hace falta saber es que el primer parámetro(k) se refiere a la clave de una entrada y el segundo(v) a su correspondiente valor

En este caso la referencia a metodos
items.forEach((k,v)->System.out::println);
da error ya efectivamente hay una única instrucción y es una llamada al método println() pero analizando los parámetros no se puede deducir que versión de println() invocar ya que hay dos parámetros k y v y println() sólo tiene un parámetro en sus versiones

Ejercicio U8_B6_e5:

Recuerda que expresiones lambda como la de este ejemplo no son habituales en la práctica. Más adelante veremos cómo conseguir este efecto de listar+filtrar con un estilo más de programación funcional.

nos puede liar un poco la sintaxis de lambda por falta de práctica. Indicamos primero una serie de cuestiones:

- Para empezar recuerda que si usas {} para el cuerpo entonces la última sentencia debe acabar con ;

quita el ; antes del } y comprueba el error
items.forEach((k,v)->{System.out.println("clave "+k+" con valor "+v);});

- por otro lado al incorporar una sentencia adicional al cuerpo lambda (el if en este caso) estoy obligado a usar {} y por tanto no me puedo olvidar del detalle arriba comentado.
- por último aquí quizá las llaves del if "enmarañan" demasiado el código y ya que se pueden evitar por incluir una única instrucción nos podemos decidir por hacerlo así

```
import java.util.HashMap;
import java.util.Map;

class App{
   public static void main(String[] args){
      Map<String, Integer> items = new HashMap<>();
          items.put("A", 10);
          items.put("B", 20);
          items.put("C", 30);
          items.put("D", 40);
          items.put("E", 50);
          items.put("F", 60);

          items.forEach((k,v)->{System.out.println("clave "+k+" con valor "+v);
                        if(k.equals("E"))
                          System.out.println("HOLA E");});
  }
}
```

con las llaves del if sobre un ;

```
import java.util.HashMap;
import java.util.Map;

class App {

   public static void main(String[] args) {
      Map<String, Integer> items = new HashMap<>();
      items.put("A", 10);
      items.put("B", 20);
      items.put("C", 30);
      items.put("D", 40);
      items.put("E", 50);
      items.put("F", 60);

      items.forEach((k, v) -> {
         System.out.println("clave " + k + " con valor " + v);
         if (k.equals("E")) {
            System.out.println("HOLA E");
         }
      });
   }
}
```

con este punto y coma extra hay error

```
items.forEach((k, v) -> {
          System.out.println("clave " + k + " con valor " + v);
          if (k.equals("E")) {
             System.out.println("HOLA E");
          }
      };);
```

a pesar de ejemplo, recuerda la observación que hicimos en otro ejercicio de los if en expresiones lambda