The `Gizmo` class represents `gadgets` that people purchase. Some `Gizmo` objects are electronic and others are not. A partial definition of the `Gizmo` class is shown below.

```
public class Gizmo
{
   /** Returns the name of the manufacturer of this Gizmo. */
   public String getMaker()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is electronic, and false
    *  otherwise.
    */
   public boolean isElectronic()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is equivalent to the Gizmo
    *  object represented by the
    *  parameter, and false otherwise.
    */
   public boolean equals(Object other)
   {
      /* implementation not shown */
   }

   // There may be instance variables, constructors, and methods not shown.
}
```

The `OnlinePurchaseManager` class manages a sequence of `Gizmo` objects that an individual has purchased from an online vendor. You will write two methods of the `OnlinePurchaseManager` class. A partial definition of the `OnlinePurchaseManager` class is shown below.

```
public class OnlinePurchaseManager
{
   /** An ArrayList of purchased Gizmo objects,
    *  instantiated in the constructor.
    */
   private ArrayList<Gizmo> purchases;

   /** Returns the number of purchased Gizmo objects that are electronic
    *  whose manufacturer is maker, as described in part (a).
    */
   public int countElectronicsByMaker(String maker)
   {
      /* to be implemented in part (a) */
   }
```

```
    /** Returns true if any pair of adjacent purchased Gizmo objects are
     *  equivalent, and false otherwise, as described in part (b).
     */
    public boolean hasAdjacentEqualPair()
    {
        /* to be implemented in part (b) */
    }

    // There may be instance variables, constructors, and methods not shown.
}
```

(a) Write the `countElectronicsByMaker` method. The method examines the `ArrayList` instance variable `purchases` to determine how many `Gizmo` objects purchased are electronic and are manufactured by `maker`.

Assume that the `OnlinePurchaseManager` object `opm` has been declared and initialized so that the `ArrayList purchases` contains `Gizmo` objects as represented in the following table.

Index in purchases	0	1	2	3	4	5
Value returned by method call `isElectronic()`	`true`	`false`	`true`	`false`	`true`	`false`
Value returned by method call `getMaker()`	`"ABC"`	`"ABC"`	`"XYZ"`	`"lmnop"`	`"ABC"`	`"ABC"`

The following table shows the value returned by some calls to `countElectronicsByMaker`.

Method Call	Return Value
`opm.countElectronicsByMaker("ABC")`	2
`opm.countElectronicsByMaker("lmnop")`	0
`opm.countElectronicsByMaker("XYZ")`	1
`opm.countElectronicsByMaker("QRP")`	0

Complete method `countElectronicsByMaker` below.

```
/** Returns the number of purchased Gizmo objects that are electronic and
 *  whose manufacturer is maker, as described in part (a).
 */
public int countElectronicsByMaker(String maker)
```

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user's purchase history to detect such occurrences and request confirmation.

Write the `hasAdjacentEqualPair` method. The method detects whether two adjacent `Gizmo` objects in `purchases` are equivalent, using the `equals` method of the `Gizmo` class. If an adjacent equivalent pair is found, the `hasAdjacentEqualPair` method returns `true`. If no such pair is found, or if `purchases` has fewer than two elements, the method returns `false`.

Complete method `hasAdjacentEqualPair` below.

```
/** Returns true if any pair of adjacent purchased Gizmo objects are
 *  equivalent, and false otherwise, as described in part (b).
 */
public boolean hasAdjacentEqualPair()
```