```
class CyberPet {
  private boolean isEating = true; // CyberPet's state
  private boolean isSleeping = false;
  private String name; // CyberPet's name

  public CyberPet() { // Constructor method
    name = "no name";
  }
  public void setName (String str) { // Access method
     name = str;
  } // setName()
  public String getName() {
     return name; // Return CyberPet's name
  } // getName()
  public void eat() { // Start eating
     isEating = true; // Change the state
     isSleeping = false;
     return;
  } // eat()
  public void sleep() { // Start sleeping
     isSleeping = true; // Change the state
     isEating = false;
     return;
  } // sleep()
  public String getState () {
     if (isEating)
        return "Eating"; // Exit the method
     if (isSleeping)
        return "Sleeping"; // Exit the method
     return "Error in State"; // Exit the method
   } // getState()
   public String toString() {
      return name + " is " + getState();
   }
} // CyberPet
public class ReferenceCall {
   public static void myMethod(CyberPet p){
     System.out.println("myMethod: pet name is " + p.getName());
     p.setName("Mary");
     System.out.println("myMethod: pet name is " + p.getName());
   } // myMethod()
   public static void main(String argv[]) {
     CyberPet pet = new CyberPet();
     pet.setName("Harry");
     System.out.println("main: pet name is " + pet.getName());
     myMethod(pet);
     System.out.println("main: pet name is " + pet.getName());
   }// main()
} // ReferenceCall
```

作成されたペットの数はmainメソッドの1行目でインスタンス化された1匹のみ。
ペットの名前について、
まずmainメソッド2行目でペットの名前を setName() により Harry に設定し、
"main: pet name is Harry"が出力された。
次にmyMethodメソッドにオブジェクトが飛ばされた中で名前が変わらないまま
"myMethod: pet name is Harry"が出力された。
その出力の後 setName() によって名前は Mary と設定され、
"myMethod: pet name is Harry"が出力された。
myMethodを終了しmainに戻り、ここでは既にmyMethodメソッドで名前は Maryと
なっているため、出力は"main: pet name is Mary" となっている。

PrimitiveCallとの違いは、PrimitiveCallでmyMethodに渡されたkがint型の変数、すなわちプリミティブ型であったのに対し、今回myMethodに渡されたpetは参照型のデータ型であり、メソッドに渡されたのがコピーされた値か参照かである。参照型の今回のデータはmyMethod内でそのメソッドを呼び直接操作された。