MORE ABOUT REFERENCE TYPES

ASHESI

Meaning of equality for reference types

- Note that when you use == to test for equality with reference types (objects), you are checking whether their references (memory addresses) are the same!

ASHESI

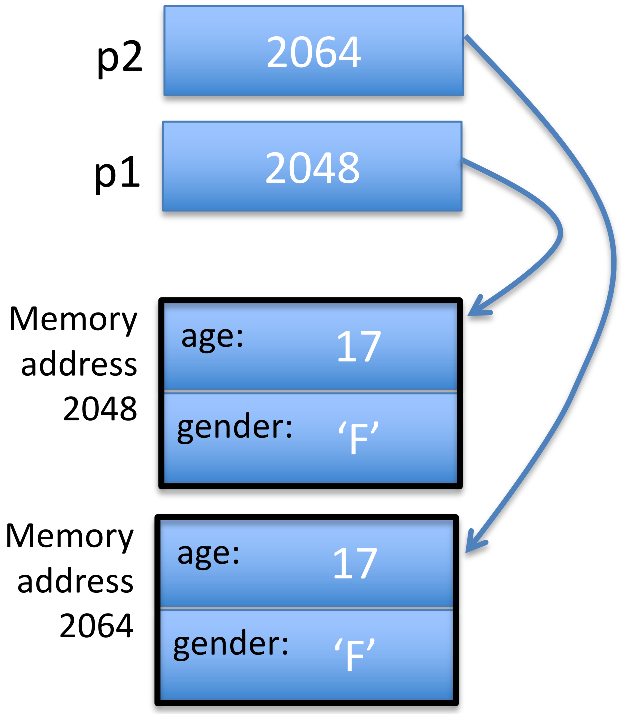
Meaning of Equality with Reference Types

```
Person p1 = new Person();
p1.init(17,'F');
Person p2 = new Person();
p2.init(17,'F');

if (p1.age == p2.age)
    System.out.println("Ages equal.");
else
    System.out.println("Ages not equal.");

if (p1.gender == p2.gender)
    System.out.println("Gender equal.");
else
    System.out.println("Gender not equal.");

if (p1 == p2)
    System.out.println("Objects equal.");
else
    System.out.println("Objects not equal.");
```



Display

```
Ages equal.
Gender equal.
Objects not equal.
```

Meaning of Equality with Reference Types

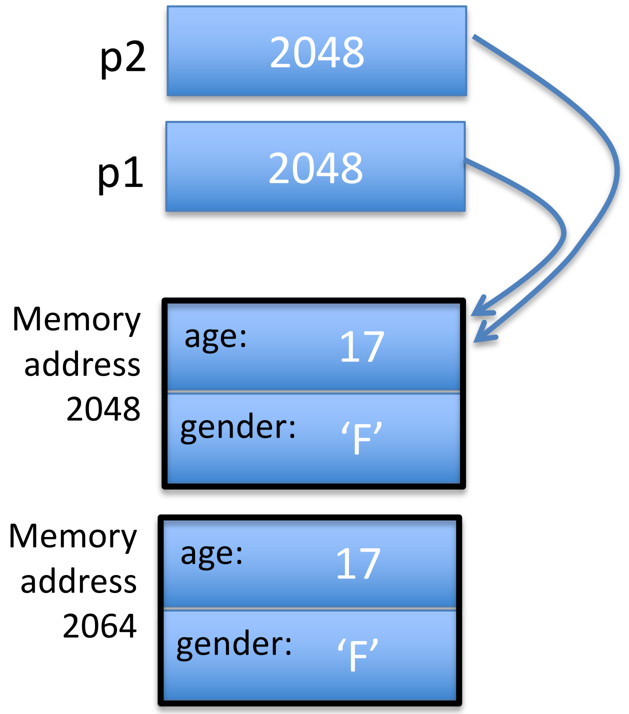
```
Person p1 = new Person();
p1.init(17,'F');
Person p2 = new Person();
p2.init(17,'F');

p2 = p1;

if (p1.age == p2.age)
    System.out.println("Ages equal.");
else
    System.out.println("Ages not equal.");

if (p1.gender == p2.gender)
    System.out.println("Gender equal.");
else
    System.out.println("Gender not equal.");

if (p1 == p2)
    System.out.println("Objects equal.");
else
    System.out.println("Objects not equal.");
```

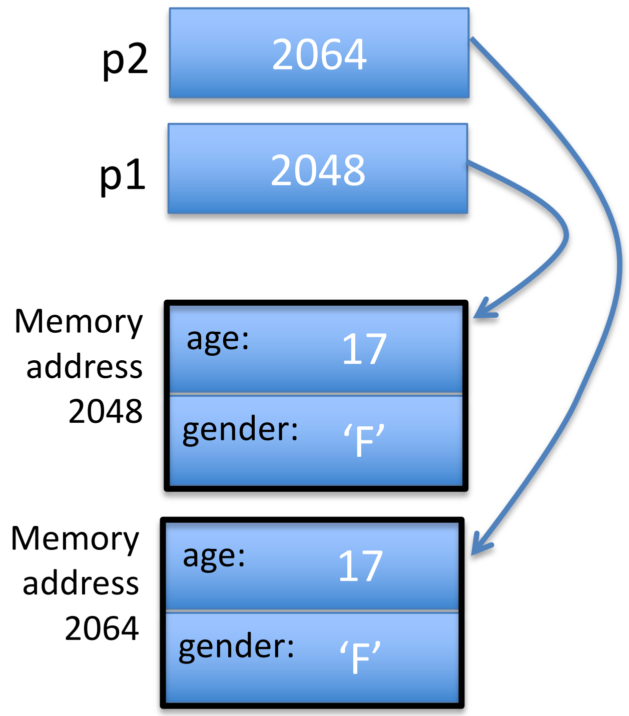


Display

```
Ages equal.
Gender equal.
Objects equal.
```

Suppose we want to check if the objects' member variables are equal?

- We can't use == for objects
- Instead, define an equals() method for your class!
  - Remember the equals() method of the String class??



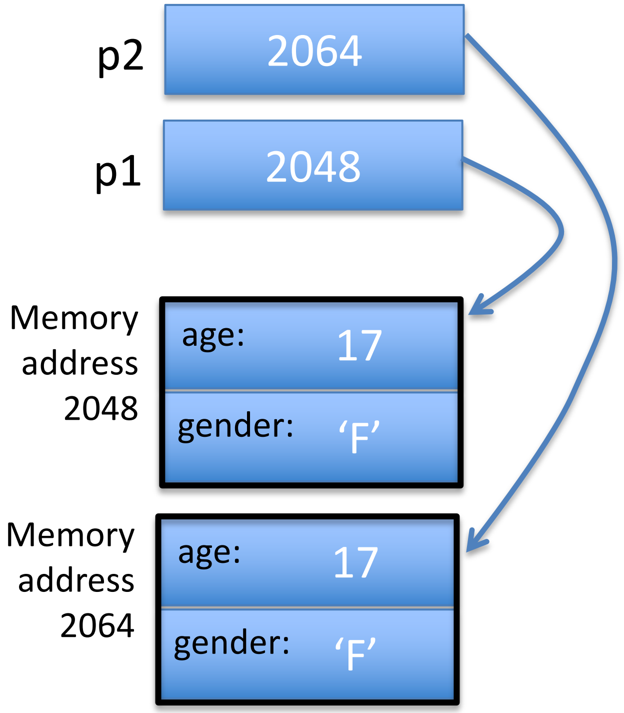
ASHESI

Testing for Equality

```
public class Person()
{
    public int age;
    public char gender;

    public void init(int a, char g){
        age = a;
        gender = g;
    }

    public boolean equals(Person other){
        if (age == other.age &&
            gender == other.gender)
            return true;
        else
            return false;
    }
}
```

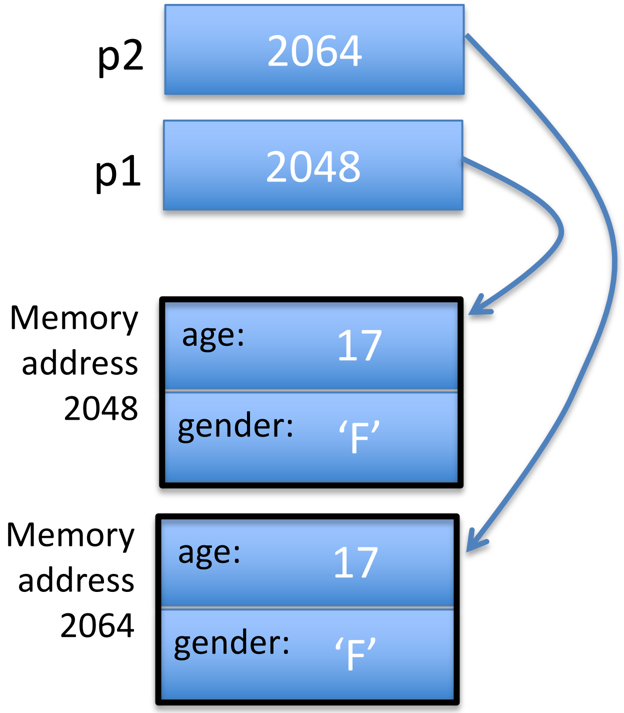


Another way of writing the same method:

```
public class Person()
{
    public int age;
    public char gender;

    public void init(int a, char g){
        age = a;
        gender = g;
    }

    public boolean equals(Person other){
        if (this.age == other.age &&
            this.gender == other.gender)
            return true;
        else
            return false;
    }
}
```



Yet another way to write the same method

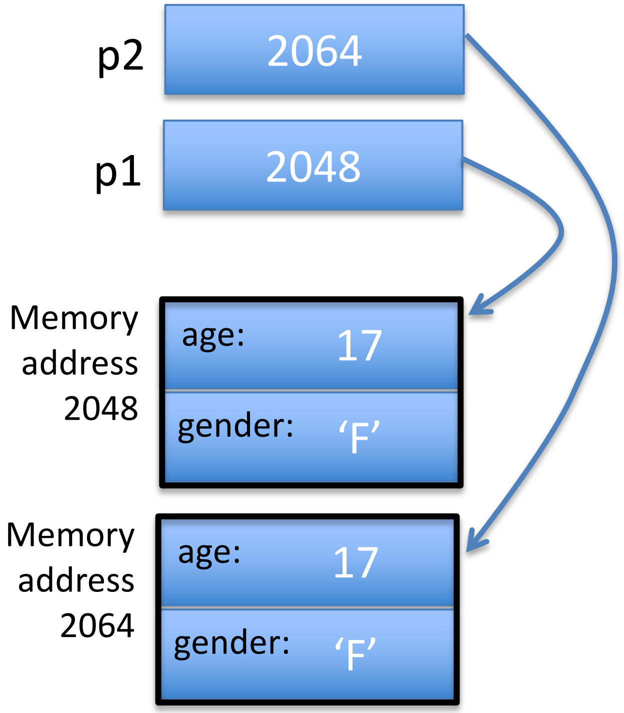
```
public class Person()
{
    public int age;
    public char gender;

    public void init(int a, char g){
        age = a;
        gender = g;
    }

    public boolean equals(Person other){
        boolean isEqual;

        isEqual = (age == other.age &&
                   gender == other.gender);

        return isEqual;
    }
}
```

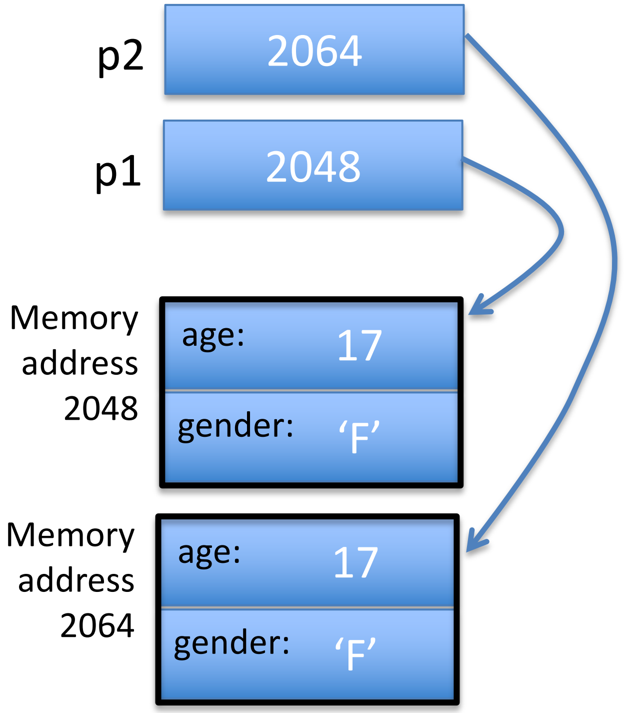


And a fourth way to write the same method:

```
public class Person()
{
    public int age;
    public char gender;

    public void init(int a, char g){
        age = a;
        gender = g;
    }

    public boolean equals(Person other){
        return (age == other.age &&
                gender == other.gender);
    }
}
```



Constructors

- A constructor is a special method that is called when you use the **new** operator to create a new object
- The purpose of a constructor is to perform initializing actions (similar to set methods)
- What is "special" about a constructor compared to other methods?
  - The name of the constructor is the same as the name of the class
  - A constructor does not have a return type
  - The **only** time a constructor can be called is when an object is first created with the keyword new. A constructor cannot be invoked on an already-created object

ASHESI

Constructors

- A default constructor has no parameters
- If you don’t define any parameters, Java will define a default parameter for you,