INFORMATION HIDING REVISITED

Ch6.5

Information Hiding Revisited

Privacy Leaks

- Instance variable of a class type contain address where that object is stored
- Assignment of class variables results in two variables pointing to same object
  - Use of method to change *either* variable, changes the actual object itself
- View insecure class, listing 6.18

```
class petPair
```

LISTING 6.18 An Insecure Class

```
/**
Class whose privacy can be breached.
*/
public class PetPair
{
    private Pet first, second;
    public PetPair(Pet firstPet, Pet secondPet)
    {
        first = firstPet;
        second = secondPet;
    }
    public Pet getFirst()
    {
        return first;
    }
    public Pet getSecond()
    {
        return second;
    }
    public void writeOutput()
    {
        System.out.println("First pet in the pair:");
        first.writeOutput();
        System.out.println("\nSecond pet in the pair:");
        second.writeOutput();
    }
}
```

LISTING 6.19 Changing a Private Object in a Class *(part 1 of 2)*

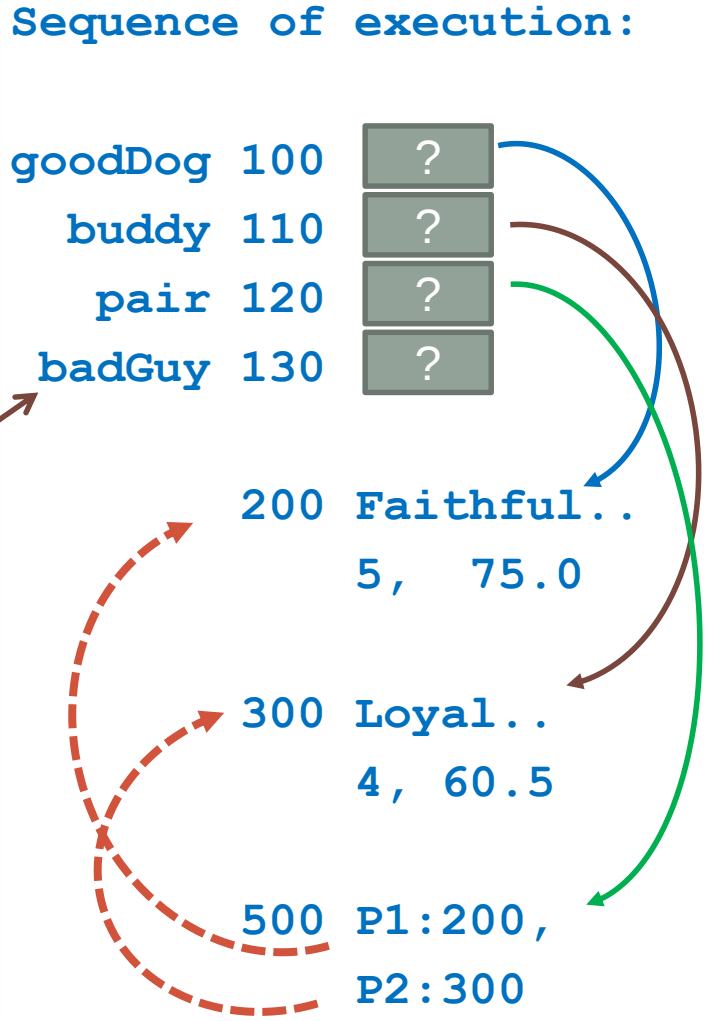
```
/**
Toy program to demonstrate how a programmer can access and
change private data in an object of the class PetPair.
*/
public class Hacker
{
    public static void main(String[] args)
    {
        Pet goodDog = new Pet("Faithful Guard Dog", 5, 75.0);
        Pet buddy = new Pet("Loyal Companion", 4, 60.5);

        PetPair pair = new PetPair(goodDog, buddy);
        System.out.println("Our pair:");
        pair.writeOutput( );

        Pet badGuy = pair.getFirst();
        badGuy.setPet("Dominion Spy", 1200, 500);

        System.out.println("\nOur pair now:");
        pair.writeOutput( );
        System.out.println("The pet wasn't so private!");
        System.out.println("Looks like a security breach.");
    }
}
```

What will this do now?



LISTING 6.19 Changing a Private Object in a Class *(part 1 of 2)*

```
/**
Toy program to demonstrate how a programmer can access and
change private data in an object of the class PetPair.
*/
public class Hacker
{
    public static void main(String[] args)
    {
        Pet goodDog = new Pet("Faithful Guard Dog", 5, 75.0);
        Pet buddy = new Pet("Loyal Companion", 4, 60.5);

        PetPair pair = new PetPair(goodDog, buddy);
        System.out.println("Our pair:");
        pair.writeOutput( );

        Pet badGuy = pair.getFirst();
        badGuy.setPet("Dominion Spy", 1200, 500);

        System.out.println("\nOur pair now:");
        pair.writeOutput( );
        System.out.println("The pet wasn't so private!");
        System.out.println("Looks like a security breach.");
    }
}
```

Sequence of execution:

```
goodDog 100   200
  buddy 110   300
   pair 120   500
 badGuy 130   200

        200 Dominion Spy
            1200, 500

        300 Loyal..
            4, 60.5

        500 P1:200,
            P2:300
```

LISTING 6.19 Changing a Private Object in a Class
(part 1 of 2)

```
/**
Toy program to demonstrate how a programmer can access and
change private data in an object of the class PetPair.
*/
public class Hacker
{
    public static void main(String[] args)
    {
        Pet goodDog = new Pet("Faithful Guard Dog", 5, 75
        Pet buddy = new Pet("Loyal Companion", 4, 60.5);

        PetPair pair = new PetPair(goodDog, buddy);
        System.out.println("Our pair:");
        pair.writeOutput( );

        Pet badGuy = pair.getFirst();
        badGuy.setPet("Dominion Spy", 1200, 500);

        System.out.println("\nOur pair now:");
        pair.writeOutput( );
        System.out.println("The pet wasn't so private!");
        System.out.println("Looks like a security breach.
    }
}
```

Sample screen output

```
Our pair:
First pet in the pair:
Name: Faithful Guard Dog
Age: 5 years
Weight: 75.0 pounds
Second pet in the pair:
Name: Loyal Companion
Age: 4 years
Weight: 60.5 pounds

Our pair now:
First pet in the pair:
Name: Dominion Spy
Age: 1200 years
Weight: 500.0 pounds
Second pet in the pair:
Name: Loyal Companion
Age: 4 years
Weight: 60.5 pounds
The pet wasn't so private!
Looks like a security breach.
```

This program has changed an object named by a private instance variable of the object pair.

Summary

- Constructor method creates, initializes object of a class
- Default constructor has no parameters
- A `static` variable shared by all objects of the class
- Divide method tasks into subtasks
- Test all methods individually
- Methods with same name, different signatures are overloaded methods