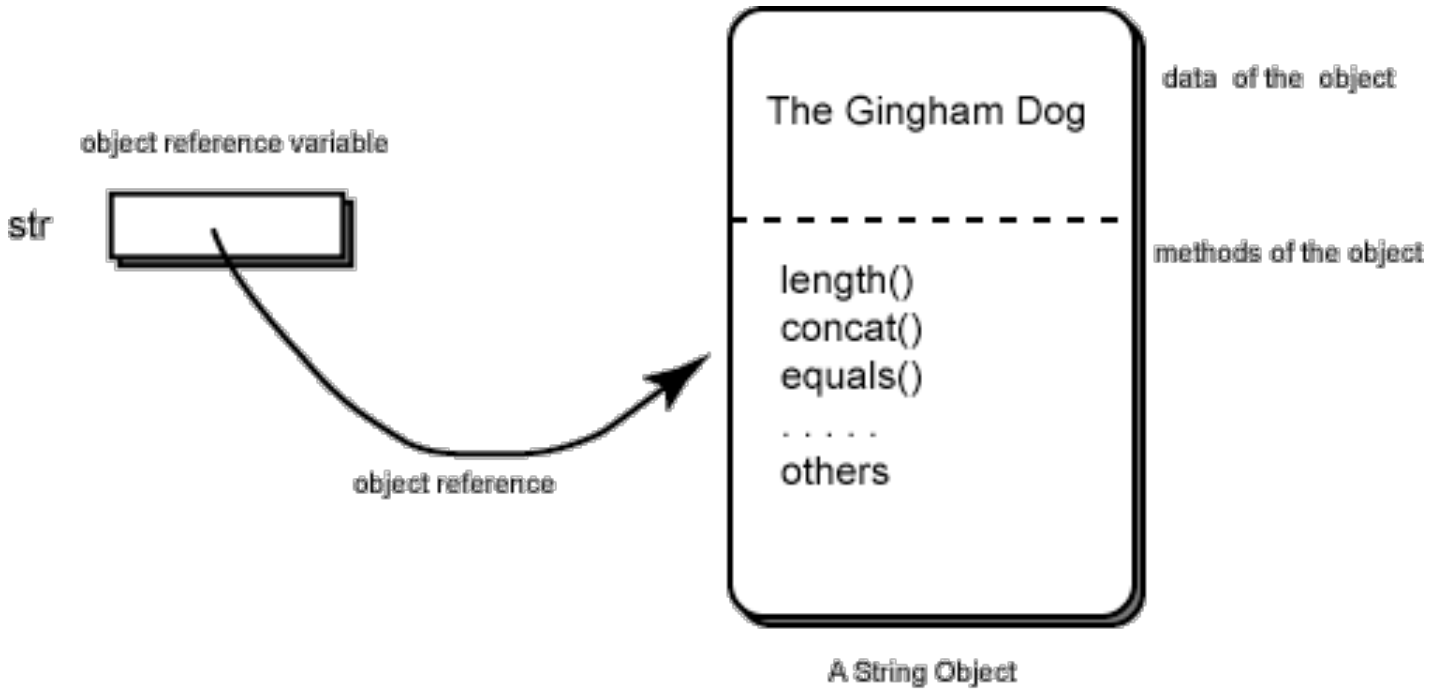
Java Summer Camp

Class 4 and 5

String str = “The Gingham Dog”;



```
String strA;  // reference to the object
String strB;  // another reference

strA = new String( “Tri-Valley" );    // Create a new object.

strB = new String( “Tri-Valley" );
```

```
if ( strA == strB )

if ( strA.equals(strB) )

if ( strA.compareTo(strB) == 0 )
```

Class Design

Requirement Gathering

Actual Design & Implementation

Checking Account

Account Number

Account Holder Name

Current Balance

Checking Account

Accept a deposit

Process a check

Check Balance

Design Summary

```
class CheckingAccount {
  // instance variables
    String accountNumber;
    String accountHolder;
    int    balance;

    //constructors
    public CheckingAccount() {

    }

    // methods
}
```

Method Overload

```
class CheckingAccount
{
  . . . .
  private int balance;

  . . . .
  public void processDeposit( int amount )
  {
    balance = balance + amount ;
  }

  public void processDeposit( int amount, int serviceCharge )
  {
    balance = balance + amount - serviceCharge;
  }

}
```

Savings Account

```
public class SavingsAccount {
  // variables

  // constructors

  // methods

}
```

States and Variables

```
public class SavingsAccount {
  // variables
    String accountNumber;
    String accountHolder;
    float  interestRate;
    float  balance;

}
```

Constructors

```
public class SavingsAccount {
  // variables

  // constructors
    public SavingsAccount(String name, String number, int initbal) {
      accountHolder = name;
      accountNumber = number;
      balance = initbal;
    }

  // methods

}
```

Methods

```
public class SavingsAccount {

    // methods
    public int checkBalance() {
      return balance;
    }

    public void acceptDeposit(int amount) {
      balance = balance + amount;
    }

    public void earnInterest(int months) {
      float interest = (interestRate/12.0)*months*balance;
      balance += interest;
    }
}
```

Challenges

- Two or More classes share the same variables, even same methods
- When something needs to change, I have to touch many places

Solutions - Inheritance

```
public class DepositAccount {
  // variables

}
```

```
public class SavingsAccount extends DepositAccount
```

Access Control

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Selection Sort

```
// The algorithm works by selecting the smallest unsorted item
// and then swapping it with the item in the next position to be filled.
public static void selectionSort(int[] data) {

}
```

Insertion Sort

```java
// Take unsort entries one at a time and then
// insert each of them into a sorted part (initially empty):
public static void insertionSort(int[] data) {

}
```

Merge Sort

```
   /** Divide the array into two (or more) subarrays
   Sort each subarray (Conquer)
   Merge them into one (in a smart way!)
   */
public static void mergeSort(int[] data) {

}
```

Binary Search

- Input array is already sorted
- Search midway between the two indices
- Determine which of the two subarrays to search
- Search midway of the chosen subarray