Arrays

- Length Fixed at Creation Time
- `int durations[] = new int [4];`
- `int durations[] = {65, 87, 72, 75;}`
- `durations[3] = 65;`
- `durations.length;`

ArrayLists

- Grow and Shrink as needed.
- As of Java 5.0 Generic, type parameter, but can not use primitive types.
- `import java.util.*;`
- `ArrayList<String> v = new ArrayList<String>();`
- `v.add(m);`
- `v.add(5,m);`
- `v.set(5,m);`
- `v.remove(5);`

ArrayLists (cont)

- `String t = v.get(5);`
- `v.size();`

ArrayLists Example

```
public class Vehicle {
 public void print() {
  System.out.println("A Vehicle");
  }
}
```

ArrayLists Example (cont)

```
public class MotorVehicle extends Vehicle {
 String regNum;
 public MotorVehicle(String no) {
 regNum = no;
 }
 public void print() {
  System.out.println
    ("A Motor Vehicle with reg no: " + regNum);
  }
}
```

ArrayLists Example (cont)

```
public class PrivateCar extends MotorVehicle {
 int numseats:
 public PrivateCar(String no, int n) {
 super(no);
 numSeats = n;
 }
 public void print() {
  super.print();
  System.out.println
    ("Private car with : "
       + numSeats + "seats");
  }
}
```

ArrayLists Example (cont)

```
public class Truck extends MotorVehicle {
 int maxL:
 public Truck(String no, int load) {
 super(no);
 maxL = load;
 }
 public void print() {
  super.print();
  System.out.println
    ("A Truck with with : " + maxL +
        "kg maximum load");
  }
}
```

ArrayLists Example (cont)

```
public class Bike extends Vehicle {
 String numGears;
 public Bike(int g) {
 numGears = g;
 }
 public void print() {
  System.out.println
    ("A bike with : " + numGears +
            " A bike with : ");
  }
}
```

Using Arrays(cont)

```
Vehicle[] veh = new Vehicle;

veh[0] = new PrivateCar("ABC123", 5);
veh[1] = new Truck("XYZ999", 10000);
veh[2] = new PrivateCar("PPP000",6);
veh[3] = new Bike(10);
```

Using Arrays(cont)

```
for (int i = 0; i <veh.length; i++)
  if (veh[i] != null){
    veh[i].print();
    System.out.println();
}
```

Using ArrayLists(cont)

```
ArrayList<Vehicle> u = new ArrayList<Vehicle>();

u.add(new PrivateCar("ABC123", 5));
u.add(new Truck("XYZ999", 10000));
u.add(new PrivateCar("PPP000",6));
u.add(new Bike(10));
```

Using ArrayLists(cont)

```
for (int i = 0; i <u.size; i++)
  if (u.get(i) != null){
    u.get(i).print();
    System.out.println();
}
```

Wrapper Classes

- `Integer`
- `Long`
- `Double`
- `Float`
- `Character`

```
Double d  = 29.95;
double x = d;

ArrayList<Double> data = new ArrayList<Double>();
data.add(29.95);
double x = data.get(0);
```

Enhanced For Loop/Java 5.0

```
double[] data = ....;
double sum = 0;
for (double e : data)
  {
    sum = sum + e;
  }

ArrayList<BankAccount> accounts = ....
double sum = 0;
for (BankAccount a: accounts)
{
   sum = sum + a.getBalance();
}
```

Object

- Every class that does not extend another class automatically extendes the class `Object`. In other words `Object` is a direct or indirect superclass of every class in Java.
- `Object` comes with several methods. These are generally overridden by the autors of other classes.
  - `String toString()` Returns a string representation of the object.
  - `boolean equals(Object otherObject)` Tests whether the object equals another object.
  - `Object clone()` Makes a full copy of the object.

toString()

```
public class BankAccount
{

private double balance;

........

 public String toString()
 {
 return "BakAccout[balance=" + balance + "]";
 }
}
```

toString() cont

```
BankAccount harrysSavings = new BankAccount(5000);
String s = monmsSavings.toString();
```

equals

```
public class Coin
{
  .....
 public boolean equals(Object other)
 {
  ...
 }
 private String name;
 private double value;
}
```

equals(cont)

```
if (coin1.equals(con2)) ,,,


if (coin1 == coin2)
```

equals (cont)

Same issue occurs with Strings

```
String string1;

String string2;

if (string1.equals(string2))


if (string1 == string2) //not useful
```

equals (cont)

```
public boolean equals(Object other)
{
  Coin other = (Coin) other;
  return name.equals(other.name) &&
                 value == other.value)

}
```

Clone

```
public class BankAccount
{
 ....
 public Object clone()
 {
   BankAccount cloned = new BankAccount();
   cloned.balance = balance;
   return cloned;
 }
}

BankAccount account2 =
    (BankAccount) account1.clone();
```

Array List Examples

```
public class Coin{
   public Coin(double aValue, String aName)
   {
      value = aValue;
      name = aName;
   }
   public double getValue()
   {
      return value;
   }
```

Array List Examples(cont)

```
public String getName() {
   return name;}
public boolean equals(Object otherObject)
  {Coin other = (Coin)otherObject;
   return name.equals(other.name)
      && value == other.value; }
private double value;
private String name;}
```

Array List Examples(cont)

```
import java.util.ArrayList;
public class Purse{
   public Purse(){
      coins = new ArrayList<Coin>();}
   public void add(Coin aCoin){
      coins.add(aCoin);}
   private ArrayList<Coin> coins;
```

Array List Examples(cont)

```
public double getTotal(){
   double total = 0;
   for (int i = 0; i < coins.size(); i++)
   {
      Coin aCoin = coins.get(i);
      total = total + aCoin.getValue();
   }
   return total;
}
```

Array List Examples(cont)

```
public int count()
{
   return coins.size();
}
```

Array List Examples(cont)

```
public boolean find(Coin aCoin)
{
   for (int i = 0; i < coins.size(); i++)
   {
      Coin c = coins.get(i);
      if (c.equals(aCoin)) return true; // found
   }
   return false; // no match in the entire array
}
```

Array List Examples(cont)

```
public int count(Coin aCoin)
{
   int matches = 0;
   for (int i = 0; i < coins.size(); i++)
   {
      Coin c = (Coin)coins.get(i);
      if (c.equals(aCoin)) matches++; // found a
   }
   return matches;
}
```

Array List Examples (cont)

```
Coin getMaximum(){
   Coin max = coins.get(0);
   for (int i = 1; i < coins.size(); i++)
   {
      Coin c = coins.get(i);
      if (c.getValue() > max.getValue())
         max = c;
   }
   return max;}}
```

OnLine Documentation

http://java.sun.com/j2se/1.5.0/docs/api/

Interface

```
public interface Comparable
{
  int compareTo(Object other);
}
```

Interface (cont)

```
public class Employee implements Comparable
{


  .......

 public int compareTo(Object other)
 {
   Employee other = (Employee) otherObject;
   if (salary < other.salary) return -1;
   if (salary > other.salary) return 1;
     return 0;
 }
  ...
```

Interface (cont)

- Can't use `new` with an interface.
- Can declare variables
- Interfaces vs Abstract classes.

```
class Employee extends Person
              implements Comarable
{

}
```

Protected Access

```
public Class BankAccount
 {

  .....

  protected double balance;


  }
```

Packages

```
package aima.search.demos;

/aima
  /search
    /demos
      /BreadthFirstDriver.java

javac aima/search/demos/BreadthFirstDriver.java
java aima.search.demos.BreadthFirstDriver
```

JAR Files

```
/home/user/classdir:.
              :home/user/archives/archive.jar
```