

Reading Input/Java 5.0

```
Scanner in = new Scanner(System.in);

System.out.print("Enter Quantity: ");
int quantity = in.nextInt();
System.out.print("Enter Price: ");
double price = in.nextDouble();

System.out.print("Enter City: ");
String city = in.nextLine(); //untill eof

System.out.print("Enter State Code: ");
String code = in.next(); // untill white space
```

Reading Input/Java 5.0 (cont)

The file name is InputTester.

```
import java.util.Scanner;
public class InputTester{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        CashRegister register = new CashRegister();
        System.out.print("Enter Price: ");
        double price = in.nextDouble();
        register.recordPurchase(price);
        System.out.print("Enter Dollars: ");
        int dollars = in.nextInt();
        System.out.print("Enter Quarters: ");
        int quarters = in.nextInt();
        System.out.print("Enter Dimes: ");
        int dimes = in.nextInt();
```

Reading Input/Java 5.0 (cont)

```
System.out.print("Enter Nickels: ");
int nickels = in.nextInt();
System.out.print("Enter Pennies: ");
int pennies = in.nextInt();
register.enterPayment(dollars,
    quarters, dimes, nickels, pennies);
System.out.print("Your Change is: ");
System.out.println(register.giveChange());
```

Inheritance

```
public class Attraction {
    // Define instance variable:
    public int minutes;
    // Define zero-parameter constructor:
    public Attraction () {
System.out.println("Calling
        Attraction constructor");
        minutes = 75;
    }
    public Attraction (int m) {minutes = m;}
```

Inheritance Continued

```
public class Movie extends Attraction {
    // Define instance variables:
    public int script, acting, direction;
    // Define zero-parameter constructor:
    public Movie () {
        System.out.println("Calling
                           Movie constructor");
        script = 5; acting = 5; direction = 5;
    }
    // Define three-parameter constructor:
    public Movie (int s, int a, int d) {
        script = s; acting = a; direction = d;
    }
}
```

Inheritance Continued

```
// Define rating:  
public int rating () {  
    return script + acting + direction;  
}  
}
```

Inheritance Continued

```
public class Symphony extends Attraction {
    // Define instance variables:
    public int music, playing, conducting;
    // Define zero-parameter constructor:
    public Symphony () {
        System.out.println("Calling
                           Symphony constructor");
        music = 5; playing = 5; conducting = 5;
    }
}
```

Inheritance Continued

```
// Define three-parameter constructor:
public Symphony (int m, int p, int c) {
    music = m; playing = p; conducting = c;
}
// Define rating:
public int rating () {
    return music + playing + conducting;
}
}
```


Abstract Classes

```
public abstract class Attraction {
    public int minutes;
    public Attraction () {minutes = 75;}
    public Attraction (int m) {minutes = m;}
    public int getMinutes () {return minutes;}
    public void setMinutes (int m) {minutes = m;}

    public abstract int rating () ;
}
```

Arrays

```
public class Demonstrate {
    public static void main (String argv[]) {
        int counter, sum = 0;
        int durations [] = {65, 87, 72, 75};
        for (counter = 0;
            counter < durations.length; ++counter)
            sum = sum + durations[counter];
        System.out.print(
            "The average of the " + durations.length);
        System.out.println(
            " duration is " + sum / durations.length);
    }
}
```

Arrays

- ▶ `int durations [];`
- ▶ `new int [4];`
- ▶ `int durations[] = new int [4];`
- ▶ `int durations[] = {65, 87, 72, 75};`
- ▶ `durations[3] = 65;`
- ▶ `durations.length;`

ArrayLists

- ▶ `import java.util.*;`
- ▶ `ArrayList v;`
- ▶ `new ArrayList() v;`
- ▶ `ArrayList v = new ArrayList();`
- ▶ `v.add(m);`
- ▶ `v.set(5,m);`
- ▶ `v.remove(5);`

ArrayLists (cont)

- ▶ `v.get(5);`
- ▶ `Foo e = (Foo) 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 u = new ArrayList();  
  
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.elementAt(i) != null){
        u.elementAt(i).print();
        System.out.println();
    }
```