ARRAYLIST

Data Types

only stores class data types
(NO INT, BOOLEAN, CHARS)
holds reference to memory

1. Size NOT FIXED
2. Built in methods

```
import java.util.ArrayList;
```

METHODS

Add
Remove

```
ArrayList list = new ArrayList();
list.add("I wanna...");
```

adds data type

```
ArrayList<String> breeds =
    new ArrayList<String>();
```

Must know for AP exam:

`List.add(e);`	adds e to end of list
`list.add(i,e);`	adds element e at location i
`List.set(i,e);`	reset element ci w/ object e
`list.get(i);`	returns element ci
`list.remove(i);`	removes element ci
`list.size();`	returns # of elements in array

CLASSES

- Class name must match file name
  - public class _____;
- All files in same file directory

- variables private in classes

Constructor

- builds class in memory

```
public classname (variable1, variable2)
{ ..... }
```

in main class

```
classname _ = new classname (variable1, variable2);
```

toString()
must be public

method to look for System.out.println();

```
public String toString()
{ return ....; }
```

VOCAB

OBJECT
- Variables — state of the object (data)
- methods — behavior of the object

CLASS
- template — DataType

ANALOGY
class - blueprint

Instantiation
- object is an instance of a class
- create a variable in memory

Constructor
- method - builds class in memory
  * ONLY ONE *

CLASSES

Sample Coin

```
public class Coin
{ private int side;
  public Coin()
    { side = 1; }
  public String toString()
    { if (side == 1)
        return "head";
      return "tail";
    }
  public void flip()
    { side = (int)(Math.random()*2);
      System.out.println(side);
    }
}
```

```
class Main
{ public static void main(String str[])
  { Coin c = new Coin();
    c.flip();
    System.out.println(c);
  }
}
```

public: can be used outside file
private: can only be used by this class

Get & Set: acessors & mutators

Constructors

No return type

Sample

```
public class Book
{ public Book(String t, String a, int y)
  { title=t;
    author=a;
  }
}
```

Mutator

changer a variable

```
public void setTitle(String t)
{ title=t; }
```

so that in the main class - call b.setTitle(); to just the title

Accessor

△ constructor to use mutator method
code not in 2 places
only place in mutator
want mutator method to do all work

this - look inside this class to see what matches

Default constructor : No parameters

```
public Book()
{ this(" ", " ", 1450); }
```

Constructor Chaining: Set values in 1 spot

Constructor: Method builds memory
Same name as class

Method overloading:
Multiple constructors (chaining) can call each other

Signature: #, order, type of parameters

Constructors

Example DIE :

```
public class Die
{   private int value;
    private int numSides;
    public Die()
    { numSides = 6;
      value = (int)(Math.random() * numSides);
    }
}
```

Default constructor
No parameters

Calling 1 constructor from another

```
public Die()
{ this(6);
}
public Die(int sides)
{ numSides = 4;
  if (sides >= 4)
    numSides = sides;
  value = (int)(Math.random() * numSides);
}
public void roll()
{ value = (int)(Math.random() * numSides) + 1;
}
```

calls the other constuc.
this must be 1st line

Constructor Chaining

Move this to new mutator constructor

Static Vs. Instance

Static

Modifier: 1 copy exists:
added to method or variable
All objects share this variable

Example: counter

```
private static int num;
```

behavior of locking num in @ class level

in constructor

```
num++;
bookID = num;
```

unique # to create a counter

↑ behind the scenes

Example: constants

```
public static final double PI = 3.14159265;
```

↓ only in this class

↓ data type

↑ can't Δ locks PI in

↑ capital letters for a constant

Example: methods

```
public static void main
```

↑ do not need to create an object to use the method

Static methods: Math.random()
pow / sqrt
ClassName.operator

Δ behavior

Adv. Classes

New techniques to build classes from scratch

INHERITANCE

3 Files Book.java
 childrensBook.java
t2_lesson07_notes.java

runner class
to check classes

Not recreating
a ton of code

↓ a child of book

```
public class childrensBook extends Book
{ private String illustrator;        ↑ locks 2 classes
  public childrensBook()
  { this ("none", "unknown", -); inherits all
  }
  public childrensBook( String t, String a,
                        String l, int g)
  { super(t,a,g);          ← calls constructor inside book MUST BE 1st
    illustrator = l;
  }
  public String toString()
  { return super.toString() +
           illustrator;
  }
```

Book: parent class
extends: links classes together
childrensBook: child class
inherits everything from parent

Extends: child class - only 1 inherit from parent
super: go up to parent and look for method

All classes inherit from OBJECT

Adv. Classes

Inheritance — Parent child relationships

Overriding Methods — Child class redefines a method from the parent

```
public class Quad
{ private int a;
  ..... }
```

parent class

```
public class Rectangle extends Quad
{  public Rectangle (int a, int b)
   { super (a, b, a, b);
   }
}
```

Child Class

Overriding — Same class

Overloading — parent class defined or child class defined

Same idea

More than 1 method with same name

Adv. Classes

Abstract Classes

parent class creates to do list for child
* cant be instantiated
* only header - no code inside

```
public abstract class Quad
{
    .
    .
    .
    public abstract int getArea();
```

any child of Quad must perform getArea

Must define getArea() method in Child class.

Child inherits abstraction from its parents

Once declared abstract - must complete all.

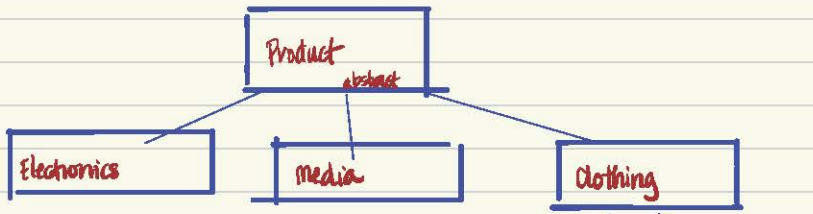
Parent can create object of type once the todo list is complete

Interfaces

Interface	"grumpy parent" must do w/o instructions ex. Comparable. Interface String s1 = "apple"; String s2 = "banana"; System.out.println(s1.compareTo(s2)); >> -1
Ultimate TODO list	compareTo object data type

Value	
less than 0	precedes
zero	same position
greater than 0	follows

Is-a has-a Rel.



will these work?

```
Product p = new Product(); NO
Clothing c = new Women(); yes
Clothing x = new Media(); no
Clothing a = new Product(); no
product p = new Clothing(); yes
```

↑ constructors

* can't instantiate an abstract class *

Is-a	Has-a
Books is a type of media	Electronics has a prize
yes add to hierarchy	dont add to hierarchy
YES EXTEND	NO EXTEND

Read code carefully. Pay attention to parent and child classes

Class Hierarchy Example

Constructor

```
public class Clothing extends product
{   public Clothing()
    {   System.out.print("B");
    }
}
.....
public class Women extends Clothing
{   public Women()
    {   * super();   <- calls clothing then moves on.
        System.out.print("A");
    }
}
```

What is output by:

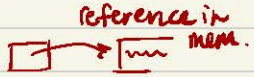
```
Women W = new Women();
```

ANS: B A

Wrapper Classes

Wrapper classes

Wraps a primitive data type to be used in ArrayLists, etc.



wrap up in an object type

* Can't store primitive in ArrayLists

Integer Wrapper

```
Integer n1 = new Integer(45);
```

Can now use methods

- compareTo

Class Integer

intValue(); returns the integer value

Double Wrapper

```
Double n1 = new Double(13.1);
```

```
Class wrapper_example
{ public static void main(String Str[])
  { Double n1 = new Double(98.232);
    ArrayList<Double> h = new ArrayList
                                <Double>();
    h.add(n1);
    h.add(new Double(0.0056));
    h.add(new Double(3.99));
    System.out.println(h);
  }
}
```