

CIS 27P

Class 4

java.text.DecimalFormat

```
import java.text.*;

class testDecimal
{
    public static void main( String [] args )
    {
        float f = (float)0.879;
        System.out.println("Value: "+f);
        DecimalFormat df = new DecimalFormat("00%");
        System.out.println("Percentage:"+df.format( f ) );
        f = 2587.4488f;
        System.out.println("Value: "+f);
        df = new DecimalFormat("$###,###.00");
        System.out.println("Currency:"+df.format(f) );
    }
}
```

OUTPUT:

Value: 0.879

Percentage:88%

Value: 2587.4487

Currency:\$2,587.45

Methods

- Method Overloading
- Supports polymorphism with type promotion
- Constructors often overloaded
- Objects can be passed as arguments as well as primitive types
- Primitive types are passed by value
- Objects are passed by reference
- Methods can return objects

Recursion

```
// A simple example of recursion.
class Factorial {
    // this is a recursive function
    int fact(int n) {
        int result;

        if(n==1) return 1;
        result = fact(n-1) * n;
        return result;
    }
}

class Recursion {
    public static void main(String args[]) {
        Factorial f = new Factorial();

        System.out.println("Factorial of 3 is "
+ f.fact(3));
        System.out.println("Factorial of 4 is "
+ f.fact(4));
        System.out.println("Factorial of 5 is "
+ f.fact(5));
    }
}
```

Access Control

- Access modifiers `public`, `private`, and `protected`.
- Applied to classes, methods, and variables.
- `public` is available to all other classes.
- `private` is available only to own class.
- `protected` is available only to inherited classes and same package.
- Not specified with a modifier is `public` within the same package.

static

- Modifier for methods and variables
- Variables are persistent at the class level
- Static methods can only call other static methods.
- Static methods can only access static variables
- Static methods can't refer to `this` or `super`
- `static` block gets executed once
- Non-static methods can call any type of method
- Non-static methods can access any type of variable
- `main()` is static because it is called before an object exists

final vars & nested classes

- `final` modifier declares variables that can't be modified
- Nested classes
 - declared inside another class
 - scope only inside outside class
 - if static, can only access static members of outer class
 - if non-static, known as inner, it can even access private members

Exercise

Write a Java class named `Student` that contains the following instance variables:

`First Name`, `Last Name`,
`Student ID`, `ClassList`

`Student` should have two constructors, one that accepts the name and id, and another that also accepts an Array of class names.

Additionally, `Student` should have a method that prints the class schedule for that student.

Inheritance

- Use `extends` keyword.
- Subclass has access to all public and protected members.
- Unmodified members in the same package are also available.
- A class can only extend one superclass.

```

// SportsCar.java
class Vehicle
{
    public int numberOfDoors;
    private String manufacturer;
    protected void printManufacturer( ) {
        System.out.println("Man: "+manufacturer);
    }
    public void setManufacturer( String in ) {
        manufacturer = in;
    }
}
class SportsCar extends Vehicle
{
    int horsepower;
    SportsCar( int hp, int nd, String man ) {
        horsepower = hp;
        numberOfDoors = nd;
        setManufacturer( man );
    }
    void printInformation() {
        printManufacturer();
        System.out.println("Horse Power:"+horsePower);
        System.out.println("Doors: "+numberOfDoors);
    }
    public static void main ( String args[] ) {
        SportsCar sc=new SportsCar(280,2,"Ferrarri");
        sc.printInformation();
    }
}

```

Inheritance (Cont.)

- A reference to a superclass can be assigned a subclass object.
 - Available members determined by reference type, not object type.
 - Subclasses can be passed as parameters when superclass is expected.
- `super` keyword

Constructors

```
class Class1
{
    Class1() { System.out.println("Class1 Default"); }
    Class1( int i ) { System.out.println("Class1 Int"); }
}
class Class2 extends Class1
{
    Class2() { System.out.println("Class2 Default"); }
    Class2( int i ) { System.out.println("Class2 int"); }
}
class Class3 extends Class2
{
    Class3() { System.out.println("Class 3 Default"); }
    Class3( int i ){
        // super(i);
        System.out.println("Class3 int");
    }
    public static void main( String [] args )
    {
        System.out.println("Default Constructor:");
        Class3 c = new Class3();
        System.out.println("Constructor with parameter:");
        Class3 c2 = new Class3( 1 );
    }
}
```

abstract & final

- Abstract methods must be implemented by subclasses.
- Classes with abstract methods must be declared abstract.
- Abstract classes can not be instantiated.
- No abstract constructors or static methods.
- Subclasses can't override final methods.
- Subclasses can't inherit final classes.
- Abstract classes can't be final.

Exceptions

- Thrown to describe an exceptional or error condition
- Can be manually thrown, or generated at run time by the virtual machine
- All exceptions must be caught somewhere
- try block
- catch block
- finally block
- uncaught exceptions
- throws

Exceptions (Cont.)

- Built-in Exceptions
- Checked vs. non-checked exceptions
- Creating your own exceptions -
inherit `java.lang.Exception`
- Use own exceptions for flow control