CS180 Recitation 2
Java classes

- Classes are the building blocks of Java applications.
- A Java application can have any number of classes.

General form of a Java class

class CLASSNAME

{  
  returntype METHODNAME1(ParameterType PARAMETERNAME1, ParameterType PARAMETERNAME2, ..)
  {
    STATEMENTS; // body of the method
  }
}
  
  returnType METHODNAME2(ParameterType PARAMETERNAME1, ParameterType PARAMETERNAME2, ..)
  {
    STATEMENTS; // body of the method
  }
}
Methods

• A class can have any number of methods.

• `main` (special method) provides the entry point for a java application where execution begins.

• Only one `main` method allowed in a java application.

• Java does not allow methods outside class definitions.

• Methods are comprised of 4 parts
  • Name of the method
  • Return type of the method
  • List of parameters
  • Body of the method
Example

class Example
{
    public static void main(String a[])
    {
        System.out.println(add(3,4));
    }
    private static int add(int num1 , int num2)
    {
        int sum;
        sum = num1 + num2;
        return sum;
    }
}
Creating objects

• In java objects are created using the new operator.

• new operator instantiates a class by allocating memory for a new object and returning a reference to that memory.

• "instantiating a class" and "creating an object" both mean the same.

Example:

Student student = new Student();

The above statement creates a new object of type Student and assigns it to the variable student.
Example

// calculates sum of 2 complex numbers of form a+ib
class Complex
{
    int real; // real part of complex number
    int imaginary; // imaginary part of complex number
}
public class Example{

    public static void main(String a[]){

        Complex complexnumber1 = new Complex();  // create instances
        Complex complexnumber2 = new Complex();
        Complex result = new Complex();

        complexnumber1.real = 10;  // assign values
        complexnumber1.imaginary = 10;
        complexnumber2.real = 20;
        complexnumber2.imaginary = 20;

        result.real = complexnumber1.real + complexnumber2.real;
        result.imaginary = complexnumber1.imaginary +
                          complexnumber2.imaginary;

        System.out.println("Result = " + result.real + "+i" +
                            result.imaginary);
    }
}
Java Standard classes

• We will look at some examples for the following standard classes in java
  • String
  • SimpleDateFormat
  • Date
  • JOptionPane
  • Scanner
  • Random
String class

• Represents character strings.
• Provides methods comparing, searching, extracting and concatenating strings.
• Another alternative to store character strings - StringBuffer class.
String examples

- Creating strings

```java
String str1 = new String(); // creates an empty string
String str2 = "cs180";
```

- String concatenation --> appends one string at the end of other

```java
String str = "cs180";
String concat = str + "spring2012" ; // concat = cs180spring2012
String concat = str.concat("spring2012"); // same as above
```

- CharAt(int index) --> returns the character at specified index

```java
char ch;
ch = "cs180".charAt(1); // ch = 's'
```

- replace(char original , char replacement)
  replaces all occurrences of original character by replacement character

```java
String s = "aaax".replace('a','b'); // s="bbbx"
```
String examples contd..

- `equals()` and `equalsIgnoreCase()`

```java
String s1 = "Hello";
String s2 = "Hello";
String s3 = "HELLO";

s1.equals(s2) --> true
s1.equals(s3) --> false
s1.equalsIgnoreCase(s3) --> true
```

- `substring(int startIndex , int endIndex )` returns a copy of substring beginning from `startIndex` until `endIndex`

```java
String s = "cs180spring2012";
String year = s.substring(11,15); // year = "2012"
```
SimpleDateFormat class

Allows defining format patterns to display date and time information.

```java
Date date = new Date(); // date holds the current date time
SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");
System.out.println(sdf.format(date)); // prints 01:02:34

SimpleDateFormat sdf = new SimpleDateFormat("dd MMM yyyy hh:mm:ss zzz"); // date month year hour min sec timezone
System.out.println(sdf.format(date)); // prints 08 Jan 2012 01:02:34 CDT

SimpleDateFormat sdf = new SimpleDateFormat("E MMM dd yyyy"); // DayOfWeek month date year
System.out.println(sdf.format(date)); // prints Wed Jan 08 2012
```
Date class

Date class encapsulates date and time.

Example

```java
Date todaysDate = new Date(); // todaysDate is initialized with current date and time

todaysDate.toString(); // prints current date and time's string representation ---> Wed Jan 18 4:05:18 EST 2012

Comparing dates

globalDate = new Date(2011, 1, 31);
globalDate2 = new Date(2012, 1, 31);
globalDate.after(globalDate2) ---> false

globalDate2.before(globalDate) ---> true

globalDate.equals(globalDate2) ---> false
```
JOptionPane class

Contains methods to display a dialog box.

- Using JoptionPane for input

  The showInputDialog() method is used user input.

```java
import javax.swing.JOptionPane;

public class InputExample {
    public static void main(String[] args) {
        String username = "";
        username = JOptionPane.showInputDialog(null, "Please enter your name: ");

        } // name entered by user is stored in the variable username
```
JOptionPane class contd..

- Using JOptionPane for output
  
  showMessageDialog() method is used to display message to user.

```java
import javax.swing.JOptionPane;

public class OutputExample {
    public static void main(String[] args) {
        String message = "Hello";
        JOptionPane.showMessageDialog(null, message);
    }
}
```
Scanner class

- Allows user to parse and read values of different types.
- Scanner breaks input into tokens using delimiter pattern (defaults to whitespace)
- Provides different next methods like nextInt(), nextFloat(), nextLong() etc to convert parsed tokens to different types.
- Provides hasNext() method to check if there are pending tokens to be read from the scanner input.
import java.util.Scanner;

public class ScannerExample {
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        int num1;
        int num2;
        System.out.print("Enter the numbers: ");
        num1 = stdin.nextInt(); // assign 1st token
        num2 = stdin.nextInt(); // assign 2nd token
        System.out.print("Sum =" + (num1+num2));
    }
}

Scanner class Example
Random class

- Generates psuedorandom numbers.
- psuedorandom since they are uniformly distributed sequences.

Example
Random random = new Random(); // uses current time as seed
int randomnum = random.nextInt();
System.out.println("Random number generated = " + randomnum);

- Seed provides the starting point for the random sequence.
- Initializing Random objects with same seed generates same random sequence.
- Using current time as seed avoids repetition of sequences.
- Other methods nextFloat(), nextLong(), nextDouble(), nextBoolean() etc available for other types of random numbers.