Reading Inputs

Reading Input from Console

- System.out refers to the standard output device and System.in to the standard input device.
- Console input is not directly supported in Java
- However, we can use the Scanner class to create an object to read input from System.in, as follows:
  - Scanner input = new Scanner(System.in);
- The object of Scanner class may invoke its methods.

Method for Scanner class object

- **nextByte()** reads an integer of the byte type.
- **nextShort()** reads an integer of the short type.
- **nextInt()** reads an integer of the int type.
- **nextLong()** reads an integer of the long type.
- **nextFloat()** reads a number of the float type.
- **nextDouble()** reads a number of the double type.
- **next()** reads a string that ends before a whitespace character.
- **nextLine()** reads a line of text (i.e., a string ending with the Enter key pressed).

Example

```
import java.util.*; // Scanner is in the java.util package

public class ConsoleInput{

	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);
		System.out.print("Enter a number for radius: ");
		double radius = input.nextDouble() ;

		double area = radius * radius * 3.14159;
		System.out.println("The area for the circle of radius " +radius + " is " + area);

	}
}
```

Another way

- To obtain a character-based stream that is attached to the console, wrap System.in in a BufferedReader object.
- **BufferedReader** supports a buffered input stream.
- To creates a BufferedReader that is connected to the keyboard:
  - BufferedReader br = new BufferedReader(
    new InputStreamReader(System.in));
- Here, br is a character-based stream that is linked to the console through System.in.

Reading Characters

```
import java.io.*;

class BRRead {
	public static void main(String args[]) throws IOException{
		char c;
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		System.out.println("Enter characters, 'q' to quit.");
		do {
			c = (char) br.read();
			System.out.println(c);
		} while(c != 'q');
	}
}
```

Reading Strings

```
import java.io.*;

class BRReadLines {
	public static void main(String args[]) throws IOException{
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		String str;
		System.out.println("Enter lines of text.");
		System.out.println("Enter 'stop' to quit.");
		do {
			str = br.readLine();
			System.out.println(str);
		}while(!str.equals("stop"));
	}
}
```

Using Command-Line Arguments

- The command-line arguments inside a Java program are stored as strings in a String array, **args** which is passed as the parameter to main( ).
- The first command-line argument is stored at args[0], the second at args[1], and so on.

```
class CommandLine {
    public static void main(String args[]) {
        for(int i=0; i<args.length; i++)
            System.out.println("args[" + i + "]: " + args[i]);
    }
}
```

Variable length argument (classical style)

```
class PassArray {
	static void vaTest(int v[]) {
		System.out.print("Number of args: " + v.length +" Contents: ");

		for(int x : v)
			System.out.print(x + " ");
			System.out.println();
		}
	}
	public static void main(String args[]){
		// array created to hold the arguments.
		int n1[] = { 10 };
		int n2[] = { 1, 2, 3 };
		int n3[] = { };
		vaTest(n1); // 1 arg
		vaTest(n2); // 3 args
		vaTest(n3); // no args
	}
}
```

Variable length argument (varargs feature)

- A variable-length argument is specified by three periods (...).
- For example, function vaTest( ) is written using a vararg:
  - static void vaTest(int ... v) {
- This syntax tells the compiler that vaTest( ) can be called with zero or more arguments.
- As a result, v is implicitly declared as an array of type int[ ].
- Thus, inside vaTest( ), v is accessed using the normal array syntax
- **The varargs parameter must be last in the argument list and there must be only one varargs parameter**.

Example varargs

```
class VarArgs {
	// vaTest() now uses a vararg.
	static void vaTest(int ... v) {
		System.out.print("Number of args: " + v.length + " Contents: ");
		for(int x : v)
			System.out.print(x + " ");
		System.out.println();
	}
	public static void main(String args[]) {
		vaTest(10);	// 1 arg
		vaTest(1, 2, 3); // 3 args
		vaTest();	// no args
	}
}
```

Another example

```
class VarArgs2 {
    static void vaTest(String msg, int ... v) {
        System.out.print(msg + v.length + " Contents: ");
        for(int x : v)
            System.out.print(x + " ");
        System.out.println();
    }

    public static void main(String args[]) {
        vaTest("One vararg: ", 10);
        vaTest("Three varargs: ", 1, 2, 3);
        vaTest("No varargs: ");
    }
}
```