Strings

Objects which are sequences of characters.

Methods – charat, equals, comparetignorecase, indexof, replace (old,new), starts with(string), subsequence(int start, end), substring(index), to char array,to lower case

ArrayList

Allows fast random read access, so you can grab any element in constant time. But adding or removing from anywhere but the end requires shifting elements. Dynamically Sized.

ArrayList<String> list = new ArrayList<String>();

Linked List

Sequential access of elements. In other words, you can walk the list forwards or backwards, but finding a position in the list takes time proportional to the size of the list. Has pointers to next link in the list which has overhead.

LinkedList<String> linkedlist = new LinkedList<String>();

ll.add(1, "A2");

Operations - add, get, set, contains, indexOf

Polymorphism

The ability of an object to take many forms.

Bird b = new Bird();

Animal a = b;

If a method parameter is of type animal, bird can also be used as a value. The ISA rule.

Inheritance

If a class extends another then it gains the fields and methods in the super class. The subclass can also add additional fields, methods and override if required.

A bird inheriting from animal is an example (bird ISA animal)

Encapsulation

Encapsulation - the technique of making the fields in a class private and providing access to the fields via public methods. The main benefit of encapsulation is the ability to modify our implemented code without breaking the code.

Interfaces

Contains method signatures and fields. An interface cannot contain an implementation of the methods.

public interface MyInterface {

public void sayHello();

}

public class MyInterfaceImpl implements MyInterface

The above class will have to implement all the methods, a class can also implement multiple methods

Abstract Class

You cannot create new instances of an abstract class. The purpose of an abstract class is to function as a base for subclass.

If a class has an abstract method, the whole class must be declared abstract. Not all methods in an abstract class have to be abstract methods.

public abstract class MyAbstractClass {

public abstract void abstractMethod();

}

public class MySubClass extends MyAbstractClass

My subclass will have to override all abstract methods.

Exceptions

Used when there is a possibility an action has potential to go wrong, the program catches the error and execution continues.

Option 1 – Throw the exception in method definition

Option 2 –

try { //try an operation

} catch (IOException e) { // show error in console

} finally { //always executes, close reader or other

}

Input

Scanner in = new Scanner(System.in);

int i = in.nextInt();

String s = in.next();

Imports

Allows use of another public package.

import java.lang.Math;

Other Math Methods

max, min, pow, round, sqrt, random

Key Words

Private – fields, methods, can only be accessed within the declared class itself, or accessed through a public getter method.

Public - fields, methods, inside a public class can be accessed from any class belonging to the Java Universe, unless in another package.

Protected - fields, methods, can only be accessed by a subclass.

Static - belong to the class instead of a specific instance, only one instance of a static field exists. Static methods cannot refer to instance fields (how does it know which one to refer to?)

Void – no value is expected to be returned

Java Pass By value

The actual parameter is *copied* into a location being used to hold the formal parameter's value during method/function execution. That location is typically a chunk of memory on the runtime stack.

Pass by reference – an alias is created in the method, changes to the alias change the actual data represented.

swap(Type arg1, Type arg2) - A test for pass by reference, if the called function swaps the values

Type temp = arg1; when execution has ended after the sub routine, PBR is supported.

arg1 = arg2;

arg2 = temp;

Casting + Parsing

Well, all casting really means is taking an Object of one particular type and “turning it into” another Object type.

String aSentenceString = (String)aSentenceObject;

Parsing

int foo = Integer.parseInt("1234")

OR Integer.toString(i)

Arrays

An array is a container object that holds a fixed number of values of a single type.

int[] anArray = { 100, 200, 300};

int[ ][ ] aryNumbers = new int[6][5]; //rows first

for(int I = 0; i<array.length;i++)

for(int I = 0; i<array[0].length;i++)

syso(array[i][j]);

Java Essentials