Féidearthachtaí as Cuimse
Infinite Possibilities

Scanner and Reference Object

Object Oriented programming

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

To date

- Classes
- Objects (Person p = new Person(“..etc”);
- Constructors
- Methods
- Method signatures
- Method overloading
- Encapsulation
- Strings

User-defined classes

- Defined our own classes:
- Use constructors
- Used methods that access and modifier member attributes

```
class Employee
{
   // Fields
   private String name;
   private double salary;
   private LocalDate hireDay;

   // Constructors
   public Employee(String n, double s, int year, int month, int day)
   {
      name = n;
      salary = s;
      hireDay = LocalDate.of(year, month, day);
   }

   // Methods
   public String getName() { return name; }
   . . .
}
```

```
public void raiseSalary(double byPercent)
{
   double raise = salary * byPercent / 100;
   salary += raise;
}
```

Member Variables Data Fields

Before you assign a value to them, They have default variables. This does not apply to local variables – you create in methods

```
public class Student {
String name; // name has default value null
int age; // age has default value 0
boolean isScienceMajor; // isScienceMajor has default value false
 char gender; // c has default value '\u0000’
}
```

Object Variables

Object variables hold a reference to an object.

To declare a reference variable, use the syntax

ClassName objectRefVar;

Book javaBook;

Create the object

ClassName objectRefVar = new ClassName();

Book javaBoook = new JavaBook();

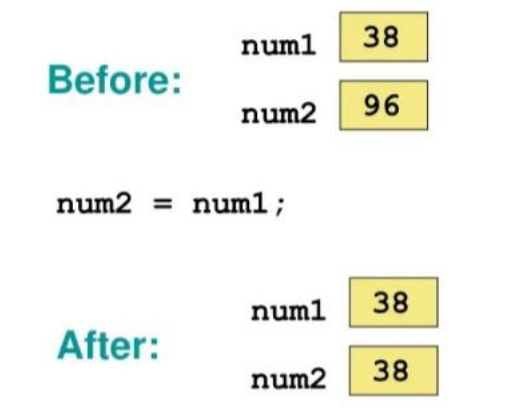
Object Variables

- Variables in java hold either a primitive value or reference to object

- NB – the variable holds a reference to the object – pointer to the location in memory

Assignment revisited

Primitive Primitive



Object reference – copy the address – not the actual object

```
Student s1 = new Student("Mary");
Student s2 = s1;      // assignment (copies reference)
s2.setName("John");
System.out.println(s1.getName());
```

Output: John
Because s1 and s2 refer to the **same object**.

```
   +------------+       +----------------+
s1 | 0x1A2F --->|------->| Student object |  name = "John"
   +------------+       +----------------+
s2 | 0x1A2F ---^
   +------------+
```

String (exception)

Once a String object is created, its contents **cannot change**. String is an object, but it behaves like a primitive in some ways.

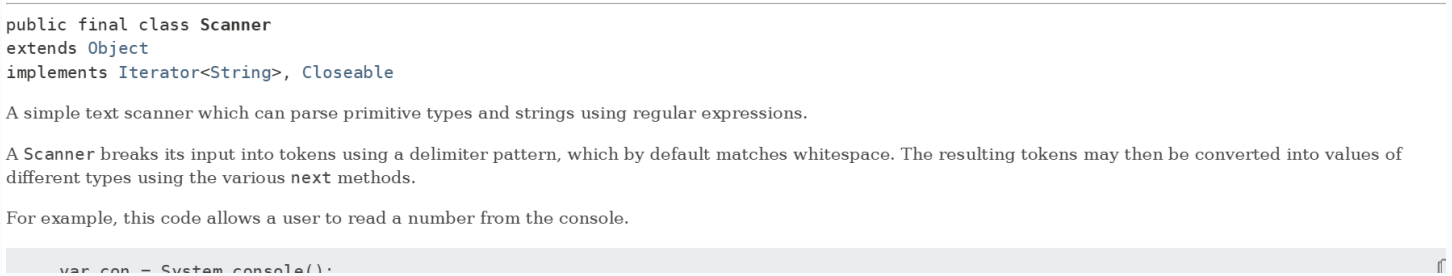
But because Strings are **immutable**, reassigning makes a **new object**.

Scanner Class – A predefined class

Scanner is part of the java.util package.

It allows a Java program to read input from different sources such as:

- Keyboard input (System.in)
- Files
- Strings



```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various `next` methods.

For example, this code allows a user to read a number from the console.

```
    var con = System.console();
```

User Input

```java
import java.util.Scanner;  // Import the Scanner class

class MyClass {
  public static void main(String[] args) {
    Scanner myObj = new Scanner(System.in);  // Create a Scanner object
    System.out.println("Enter username");

    String userName = myObj.nextLine();  // Read user input
    System.out.println("Username is: " + userName);  // Output user input
  }
}
```

Scanner Class Methods

Method	Description
nextBoolean()	Reads a boolean value from the user
nextByte()	Reads a byte value from the user
nextDouble()	Reads a double value from the user
nextFloat()	Reads a float value from the user
nextInt()	Reads a int value from the user
nextLine() next()	Reads a String value from the user
nextLong()	Reads a long value from the user
nextShort()	Reads a short value from the user

Demo (Code on Brightspace: lab1 solution)

```java
package ie.tudublin.cmpu2016.lab1;
import java.util.Scanner;

public class TemperatureConverter {

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            for (int i = 0; i < 3; i++) {
                System.out.print("Enter Celsius: ");
                double c = sc.nextDouble();
                double f = (c * 9.0 / 5.0) + 32.0;
                System.out.println("Fahrenheit = " + f);
            }
            sc.close();
        }
    }
```

Lab Code (Code on Brightspace)

```java
package Lab3;

//Final version for Parts 2 & 3 (encapsulation + validation + playTrailer)

public class Movie {
 // 1) Encapsulated attributes
 private String title;
 private String genre;
 private int durationMinutes;

 // 2) Constructor uses setters (no backdoor access)
 public Movie(String title, String genre, int durationMinutes) {
     setTitle(title);
     setGenre(genre);
     setDurationMinutes(durationMinutes);  // validation happens here
 }

 // 3) Getters
 public String getTitle() {
     return title;
 }

 public String getGenre() {
     return genre;
 }

 public int getDurationMinutes() {
     return durationMinutes;
 }

 // 4) Setters (with validation for duration)
 public void setTitle(String title) {
     // simple gatekeeping
     if (title == null || title.trim().isEmpty()) {
         System.out.println("Invalid title. Keeping previous value.");
         return;
     }
     //only expect the below from students
     this.title = title.trim();
 }

 public void setGenre(String genre) {
     if (genre == null || genre.trim().isEmpty()) {
         System.out.println("Invalid genre. Keeping previous value.");
         return;
     }
     //again only expecting the below from students or just direct assignment
```