CB 714
Advanced System Analysis

Dr. Mohamed Saeid Eid

Spring - 2018

Introduction to Java Programming

- What is Java?

- Where you would find Java language?

- Pros of Java language

Outline

- Basic introduction to Java language
- First code: “Hello World”
- Primitive Data types
- Conditions and Loops
- Data Structures
- Functions and Methods
- Object Oriented Programming

Setup

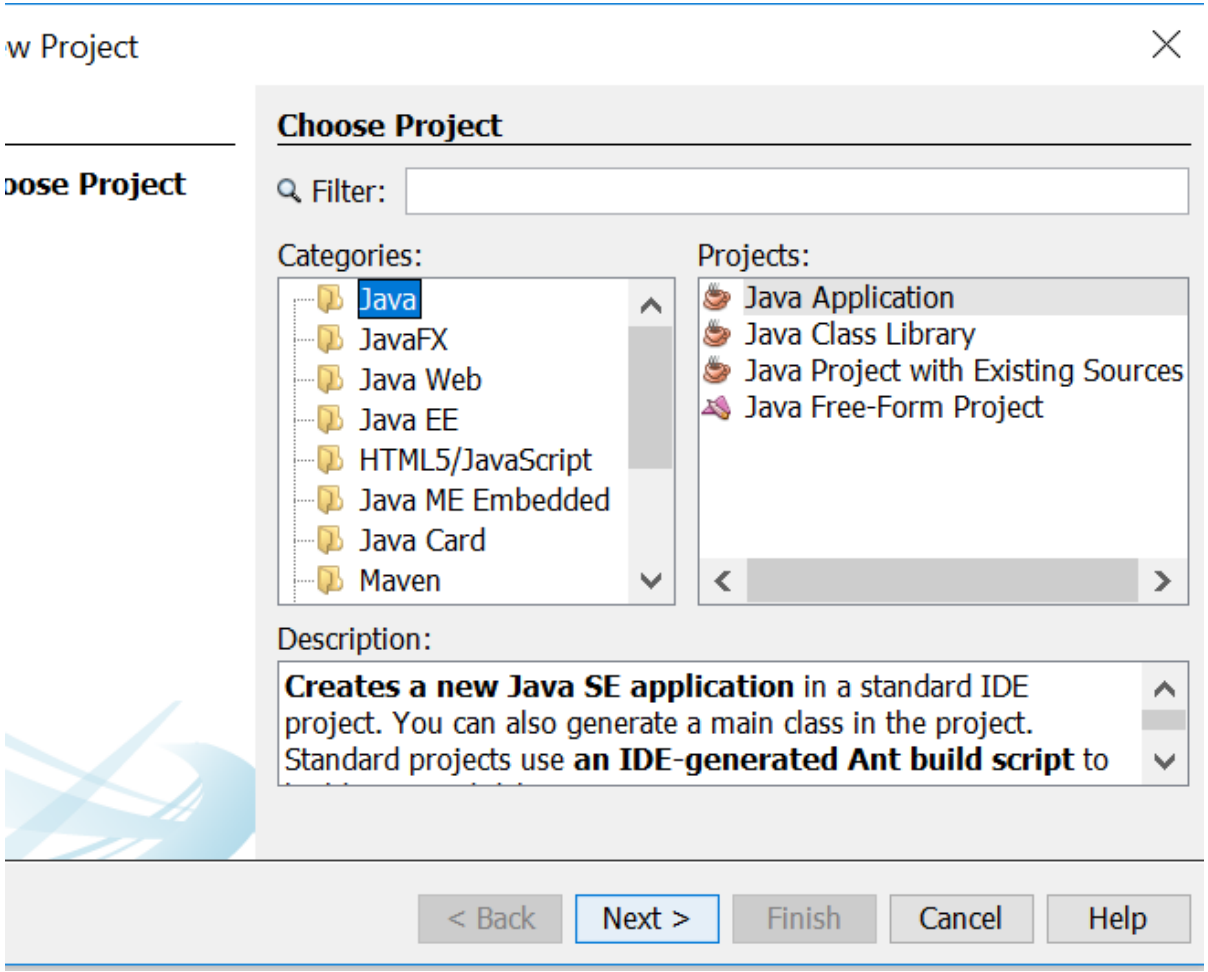
- Install the latest JDK
- Recommended Platform: Netbeans 8.1 (Free)

Java 101

- Java is an Object Oriented Programming Language (more about that later).
- There is a structure to follow to create a code
- Project>classes>methods
- Libraries

Java 101

- Creating your first project: Hello World
- Using Netbeans, go to File>New Project
- Here you can select the type of programming language.
- We are using Java, and Java Application



Hello World

```java
package tutorials;

/**
 *
 * @author Saeid
 */
public class Tutorials {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World");
    }

}
```

Hello World

- “System” as an embedded classes/library within Java to help you control some basic function

- “System.out” has various of output features to the main counsel interface (NetBeans Platform)

Primitive Java Data Type

- All variables must be declared before using them.

- This saves some part of the memory for the variables being declared and used.

Primitive Java Data Type

Variable	Size
byte	8-bit number
short	16-bit number
int	32-bit number
long	64-bit number
float	32-bit decimal precession
double	64-bit decimal precession
char	16-bit Unicode (for one character)
String	Number of char to make a word or sentences
boolean	True or false logic variable

Adjusting Hello World

- Creating a String for “Hello World”
- Adding it to the “System printout function”
- Adding another sentence for your name and age. Choose appropriate data type

Operations

- Logic
  - If
  - Switch
- Loops
  - For
  - Do-while

If function

- Logic function (if) is a powerful tool to control your program

```
if (condition)
{ statement;}
else {statement;}
```

If function

- Assume we have a two integer variables (i.e.; 4 and 15). And we know the answer for their summation (19).
- We can create a code that asks the user to add in a value for the answer and check if it is correct or wrong.

- Start with knowing the answer, then develop a code that can read the answer

If condition

```
public static void main(String[] args)  {

    int x1 = 4;
    int x2 = 15;
    int modelAnswer = 19;

    System.out.println("Please enter the sum of " + x1 + " and " +x2);

    Scanner read = new Scanner(System.in);

    int answer = read.nextInt();

    System.out.println("you entered " + answer);
    if (answer==modelAnswer){
        System.out.println("Correct Answer");
    }else {
        System.out.println("Wrong Answer");
    }

}
```

Nested if and ladder if

```
if (condition)
{ if (condition) {statement;
else statement;}
}
else {statement;}
```

Switch

- Switch is an operator that lets you choose a specific action for each case

```
switch(expression){
    case value1:
     statement
       break;
    case value2:
     statement
       break;
        .
        .
        .
      default:
     statement
        }
```

for loop

- Simply, a loop to perform a task several times

for (*initialization; termination condition; iteration step*)

{

statement;

}

Summation example

- Get the sum of integers from 1 to n (for example 1 to 10)

```
int sum = 0;
for (int i = 0; i<=10; i++)
{
    sum=sum+i;
}
System.out.println("sum = " + sum);
```

Nested for loop

```
for (initialization; termination condition; iteration step)
                                {
for (initialization; termination condition; iteration step)
                                {
                            statement;
                                }
                                }
```

do-while loop

- do-while loop is a loop function that will repeat till a condition is met

do

{

statement;

}while (*condition is false*)

do-while loop

```
boolean loopCheck = false;
int Sum = 0;
do{
    Sum++;
    if (Sum ==55) loopCheck = true;
}while (!loopCheck);
System.out.println("Sum = " + Sum);
```

Classes

- A class is a template that defines the form of an object. It is the abstraction of the object itself.

- Java uses a class to create an *object*. Objects are then just an *instance* of that abstraction (class) that holds all the major attributes of the class.

Classes – declare, allocate, use

```
class car{
double engineCapacity;
int numDoors;
double KPL;
}
```

```
class twoCars{
  public static void main (String args[]){

  car BMW320 = new car();
  BMW.engineCapacity= 2.0;
  BMW.numDoors = 4;

  car ToyotaCorolla = new car();
  ToyotaCorolla.engineCapacity = 1.6;
  ToyotaCorolla.numDoors = 4;
        }
}
```

Methods and Classes

- Methods are processes or subroutines that change the variables within an object or other objects.

- a method has a name, type, parameter(s) and body

```
type methodName(parameters){
//body of method
}
```

- Remember System.out.println(); class.subclass.method;

Methods and Classes

- A method type can be void; no value to return

```
public void setKPL(){
        KPL = 3;
        }
```

Methods and Classes

- A method can also return a value

```
class car{
double engineCapacity;
int numDoors;
double KPL;

double setKPL (){
        return 6.5;
        }
}
```

```
class twoCars{
  public static void main (String args[]){

  car BMW320 = new car();
  BMW.engineCapacity= 2.0;
  BMW.numDoors = 4;
        BMW.KPL = BMW.setKPL();

  car ToyotaCorolla = new car();
  ToyotaCorolla.engineCapacity = 1.6;
  ToyotaCorolla.numDoors = 4;
        }
}
```

Methods and Classes

- Method parameters are the input of the method

```
class car{
double engineCapacity;
int numDoors;
double KPL;

double setKPL (double distance, double tankSize){
        return distance/tankSize;
        }
}
```

```
class twoCars{
  public static void main (String args[]){

  car BMW320 = new car();
  BMW.engineCapacity= 2.0;
  BMW.numDoors = 4;
        BMW.KPL = BMW.setKPL(500,60);

        }
}
```

Class constructor

- A class constructor is a method that initializes the object.
- It has the same name of the class and no return type.
- Java automatically creates it, but can help you in creating a better code.
- The parameters of the class constructor help in creating the object needed.

Class constructor

```
class car{
double engineCapacity;
int numDoors;
double KPL;

car(double c, int d){
        engineCapacity = c;
        numDoors = d;
        }
}
```

```
class twoCars{
   public static void main (String
args[]){

   car BMW320 = new car(2.0, 4);
   car ToyotaCorolla = new car(1.6,4);

          }
}
```

Reflection on Object Oriented Programming

- In OOP, we can build our codes as reusable abstract blocks.

- After creating the object from a class (with or without constructor) we can control the object through the methods within it

- Remember, we only scratched the surface, you might want to learn about public and private variables and methods; data structures like Arrays and Lists

Recursive Methods

- Recursion is the process of repeating the same method till you find a solution.

- Recursion is found in different mathematical formulas (factorial, summation of integers, etc.)

Recursive Methods

```
int function (int variable){
if (variable == termination) return termination value;
else return function(variable*);
}
```

Recursive Methods

- Factorial is a perfect example for recursive
- Factorial is the multiplication of value n by n-1, till you reach 1.

- Try this on your PC using a recursive method