B. Sc. (H) Computer Science Semester II
BHCS03 – Programming in JAVA

Topic:

- **A Closer Look at Methods and Classes**
- **Inheritance**

Command Line Arguments

The java command-line argument is an argument i.e. passed at the time of running the java program.

The arguments passed from the console can be received in the java program and it can be used as an input.

So, it provides a convenient way to check the behavior of the program for the different values. You can pass N (1,2,3 and so on) numbers of arguments from the command prompt.

Simple example of command-line argument

In this example, we are receiving only one argument and printing it.

To run this java program, you must pass at least one argument from the command prompt.

```
class CommandLineExample{
public static void main(String args[]){
System.out.println("Your first argument is: "+args[0]);
}
}
```

compile by > javac CommandLineExample.java
run by > java CommandLineExample sonoo
Output: Your first argument is: sonoo

Varargs: Variable length arguments

A method with variable length arguments(Varargs) in Java can have zero or multiple arguments. Variable length arguments are most useful when the number of arguments to be passed to the method is not known beforehand. They also reduce the code as overloaded methods are not required.

Example

```
public class Demo {

  public static void Varargs(String... str) {
```

```
    System.out.println("\nNumber of arguments are: " + str.length);
    System.out.println("The argument values are: ");
    for (String s : str)
      System.out.println(s);
  }
  public static void main(String args[]) {
    Varargs("Apple", "Mango", "Pear");
    Varargs();
    Varargs("Magic");
  }
}
```

Output

```
Number of arguments are: 3
The argument values are:
Apple
Mango
Pear

Number of arguments are: 0
The argument values are:

Number of arguments are: 1
The argument values are:
Magic
```

Overloading Varargs Methods

Overloading allows different methods to have same name, but different signatures where signature can differ by number of input parameters or type of input parameters or both.

```
public class Main {
  static void vaTest(int ... no) {
    System.out.print(
      "vaTest(int ...): " + "Number of args: " + no.length +" Contents: ");
    for(int n : no)System.out.print(n + " ");
```

```
    System.out.println();
  }
  static void vaTest(boolean ... bl) {
    System.out.print( "vaTest(boolean ...) " + "Number of args: " + bl.length + " Contents: ");
    for(boolean b : bl)
            System.out.print(b + " ");
             System.out.println();
  }
  static void vaTest(String msg, int ... no) {
    System.out.print( "vaTest(String, int ...): " + msg +"no. of arguments: "+ no.length +" Contents: ");
    for(int  n : no)
             System.out.print(n + " ");
            System.out.println();
  }
  public static void main(String args[]) {
    vaTest(1, 2, 3);
    vaTest("Testing: ", 10, 20);
    vaTest(true, false, false);
  }
}
```

Result

The above code sample will produce the following result.

```
vaTest(int ...): Number of args: 3 Contents: 1 2 3
vaTest(String, int ...): Testing: no. of arguments: 2
Contents: 10 20
vaTest(boolean ...) Number of args: 3 Contents:
true false false
```

Inheritance

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviour of a parent object. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class.

extends Keyword

extends is the keyword used to inherit the properties of a class. Following is the syntax of extends keyword.

Syntax

```
class Super {
  .....
  .....
}
class Sub extends Super {
```

```
  .....
  .....
}

class Calculation {
  int z;

  public void addition(int x, int y) {
    z = x + y;
    System.out.println("The sum of the given numbers:"+z);
  }

  public void Subtraction(int x, int y) {
    z = x - y;
    System.out.println("The difference between the given numbers:"+z);
  }
}

public class My_Calculation extends Calculation {
  public void multiplication(int x, int y) {
    z = x * y;
    System.out.println("The product of the given numbers:"+z);
  }

  public static void main(String args[]) {
    int a = 20, b = 10;
    My_Calculation demo = new My_Calculation();
    demo.addition(a, b);
    demo.Subtraction(a, b);
    demo.multiplication(a, b);
  }
}
```

Compile and execute the above code as shown below.

```
javac My_Calculation.java
java My_Calculation
```

After executing the program, it will produce the following result –

Output

```
The sum of the given numbers:30
The difference between the given numbers:10
The product of the given numbers:200
```

The super keyword

The super keyword is similar to this keyword. Following are the scenarios where the super keyword is used.

- It is used to differentiate the members of superclass from the members of subclass, if they have same names.
- It is used to invoke the superclass constructor from subclass.

Differentiating the Members

If a class is inheriting the properties of another class. And if the members of the superclass have the names same as the sub class, to differentiate these variables we use super keyword as shown below.

```
super.variable
super.method();
```

```
class Super_class {
  int num = 20;

  // display method of superclass
  public void display() {
    System.out.println("This is the display method of superclass");
  }
}

public class Sub_class extends Super_class {
  int num = 10;

  // display method of sub class
  public void display() {
    System.out.println("This is the display method of subclass");
  }

  public void my_method() {
    // Instantiating subclass
    Sub_class sub = new Sub_class();

    // Invoking the display() method of sub class
    sub.display();

    // Invoking the display() method of superclass
    super.display();

    // printing the value of variable num of subclass
    System.out.println("value of the variable named num in sub class:"+ sub.num);

    // printing the value of variable num of superclass
    System.out.println("value of the variable named num in super class:"+ super.num);
  }

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

```
    Sub_class obj = new Sub_class();
    obj.my_method();
  }
}
```

Compile and execute the above code using the following syntax.

```
javac Super_Demo
java Super
```

On executing the program, you will get the following result –

Output

```
This is the display method of subclass
This is the display method of superclass
value of the variable named num in sub class:10
value of the variable named num in super class:20
```

Invoking Superclass Constructor

If a class is inheriting the properties of another class, the subclass automatically acquires the default constructor of the superclass. But if you want to call a parameterized constructor of the superclass, you need to use the super keyword as shown below.

```
super(values);
```

Example

```
class Superclass {
  int age;

  Superclass(int age) {
    this.age = age;
  }

  public void getAge() {
    System.out.println("The value of the variable named age in super class is: " +age);
  }
}

public class Subclass extends Superclass {
  Subclass(int age) {
    super(age);
  }

  public static void main(String argd[]) {
    Subclass s = new Subclass(24);
    s.getAge();
  }
}
```

Compile and execute the above code using the following syntax.

```
javac Subclass
java Subclass
```

On executing the program, you will get the following result –

Output

```
The value of the variable named age in super class is: 24
```

Method Overriding in Java

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java. In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

```
class Human{
  //Overridden method
  public void eat()
  {
    System.out.println("Human is eating");
  }
}
class Boy extends Human{
  //Overriding method
  public void eat(){
    System.out.println("Boy is eating");
  }
  public static void main( String args[]) {
    Boy obj = new Boy();
    //This will call the child class version of eat()
    obj.eat();
  }
}
```

Output:

```
Boy is eating
```

Advantage of method overriding

The main advantage of method overriding is that the class can give its own specific implementation to a inherited method without even modifying the parent class code.

This is helpful when a class has several child classes, so if a child class needs to use the parent class method, it can use it and the other classes that want to have different implementation can use overriding feature to make changes without touching the parent class code.

Method Overriding and Dynamic Method Dispatch

Method Overriding is an example of runtime polymorphism. When a parent class reference points to the child class object then the call to the overridden method is determined at runtime, because during method call which method(parent class or child class) is to be executed is determined by the type of object. This process in which call to the overridden method is resolved at runtime is known as dynamic method dispatch. Let's see an example to understand this:

```
class ABC{
  //Overridden method
  public void disp()
  {
            System.out.println("disp() method of parent class");
  }
}
class Demo extends ABC{
  //Overriding method
  public void disp(){
            System.out.println("disp() method of Child class");
  }
  public void newMethod(){
            System.out.println("new method of child class");
  }
  public static void main( String args[]) {
            ABC obj = new ABC();
            obj.disp();
            ABC obj2 = new Demo();
            obj2.disp();
  }
}
```

Output:

```
disp() method of parent class
disp() method of Child class
```

Abstract Methods and Classes

An *abstract class* is a class that is declared abstract—it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed.

An *abstract method* is a method that is declared without an implementation (without braces, and followed by a semicolon), like this:

```
abstract void moveTo(double deltaX, double deltaY);
```

If a class includes abstract methods, then the class itself *must* be declared abstract, as in:

```
public abstract class GraphicObject {
  // declare fields
  // declare nonabstract methods
  abstract void draw();
}
```

When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class. However, if it does not, then the subclass must also be declared abstract.

```
abstract class Shape{
abstract void draw();
}
//In real scenario, implementation is provided by others i.e. unknown by end user
class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}
class Circle1 extends Shape{
void draw(){System.out.println("drawing circle");}
}
//In real scenario, method is called by programmer or user
class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle1();//In a real scenario, object is provided through method, e.g., getShape() method
s.draw();
}
}
```

Output:

```
drawing circle
```

final keyword with inheritance in java

- The final keyword is final that is we cannot change.
- We can use final keywords for variables, methods, and class.
- If we use the final keyword for the inheritance that is if we declare any method with the final keyword in the base class so the implementation of the final method will be the same as in derived class.
- We can declare the final method in any subclass for which we want that if any other class extends this subclass.

Note: Please solve examples on the topics discussed above from reference book [*].

*Schildt, H. Java: The Complete Reference. 9th edition. McGraw-Hill Education.

References/Resources:

1. Balaguruswamy, E. (2014). Programming with JAVA: A Primer. 5th edition. India: McGraw Hill Education
2. Horstmann, C. S. (2017). Core Java - Vol. I – Fundamentals (Vol. 10). Pearson Education
3. https://www.tutorialspoint.com
4. https://beginnersbook.com
5. https://docs.oracle.com
6. https://www.includehelp.com
7. Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.

Assignment

Q1. What will be the output of the below program?

a)

```
class A
{
   public A()
   {
      System.out.println("Class A Constructor");
   }
}

class B extends A
{
   public B()
   {
      System.out.println("Class B Constructor");
   }
}

class C extends B
{
   public C()
   {
      System.out.println("Class C Constructor");
   }
```

```
}

public class MainClass
{
   public static void main(String[] args)
   {
      C c = new C();
   }
}
```

b)

```
class A
{
   String s = "Class A";
}

class B extends A
{
   String s = "Class B";

   {
      System.out.println(super.s);
   }
}

class C extends B
{
   String s = "Class C";

   {
      System.out.println(super.s);
   }
}

public class MainClass
{
   public static void main(String[] args)
   {
      C c = new C();

      System.out.println(c.s);
   }
}
```

c)

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

```
for(int i=0; i<args.length; i++)
System.out.println("args[" + i + "]: " + args[i]);
}
}
```

Q2. Can abstract class have constructors in Java?

Q3. Create an abstract class 'Parent' with a method 'message'. It has two subclasses each having a method with the same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message' by creating an object for each subclass.

Q4. An abstract class has a construtor which prints "This is constructor of abstract class", an abstract method named 'a_method' and a non-abstract method which prints "This is a normal method of abstract class". A class 'SubClass' inherits the abstract class and has a method named 'a_method' which prints "This is abstract method". Now create an object of 'SubClass' and call the abstract method and the non-abstract method. (Analyse the result)

Q5. Write a java code to find whether a number is prime or not where number is accepted from command line.