25 Aug. Completed functions

Page No.
Date

Nested switch.

→ Inside another switch.

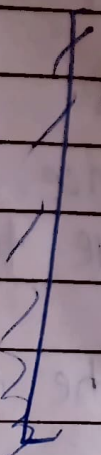
* ## Functions Of Java

Function / methods in Java.

Functions / methods (in Java) :-

- A method is a block of code which only runs when it is called.
- To reuse code : define the code once & use it many times

Syntax



Syntax

this method mymethod() does not have a return value

name of method

```
public class main {
    static void mymethod() {
        // code
    }
}
```

```
public class main {
    access-modifier return-type method()
    {
        // code
        return statement; → f^n ends here
    }
}
```

- method() → calling the function
  - name of function.

- return-type :- A return type statment causes the program control to transfer back to the caller of a method.

A return type may be primitive type like int, char or void type (returns nothing).

Example:-

```
public class Greeting {

    public static void main(String [] args){
        greeting();
    }
    static void greeting(){
        System.out.println("Hello World");
    }
}
```

ex

```
public class passinger {
public static void main(String[] args){
    String chacha = "Iron Man";
    greet(chacha);
    }
static void greet(String naam){
        System.out.println(naam);
    }
}
```

Practice Today task — function completed with 10 examples

Page No.
Date

② primitives :- int, short, char, byte
↓
passing value

★ objects ⊙ stay :- passing value of the reference.

⟺ can't use reference variable again in block scope.

★ Scope Imp topic

* 1st :- function scope.

ex
```
public class scope{
    public static void main(String[] args)
{
        int a=10;
        int b = 20;
}
static void random(){
int num = 67;
System.out.println(num);
}
```

Intmain

Scope — you can only this things in inside the functions.

Page No.
Date

* Block scope

- [you can not initialize again.]
- values initialize in this block remain in blocks
- can not use outside the block
- you can not initialize again, but can change the value

```
int a = 100   [initializing]
{
  a = 25;     [updating]
}
```

⟹ oreginal int change

* Scoping in for loop
- change inside the for loop
- Cannot initialize again over.

anything initialize the outside the box you can ~~able~~ auaible inside the box

Static → object independent.

Page No.
Date

Shadowing

- using two variables with same name with the scope is overlap.

```
public class shadowing {
    static int x = 90;
    // this will be shadowed at line 4
public static void main (String[] args) {
    System.out.println(x); // 90
    int x = 40; // class variable at line 4 is shadow
    System.out.println(x); // 40
    fun();
}

static void fun() {
    System.out.println(x);
}
}
```

→ scope will begin when - value is initialized

* Variable nth argument

```
public class VarArgs {
public static void main(String[] args) {
    fun(...v: 2,3,4,5,56,87,23,45,65);
}

static void fun(int...v) {
System.out.println(Arrays.toString(v));
}
}
```

output :-

[2,3,4,5,56,87,23,45,65]

main {

multiple(a:2, b:2, ...v:"kunal")
}

static void mutiple(int a;int b; String){
}

order is Important

* Function Overloading.

⟹ two or more functions of same name but different arguments.

ex

```
public class overloading{
public static void main(String[] args){
  fun (a:67);
  fun (name:"kunal");
}

static void fun(int a){
    system.out.println(a);
}

static void fun (string name){
  System.out.println(name);
}
}
```

program – to check the number is prime or not.

```
public class Questions {
    public static void main(String[] ar
        Scanner sn = new Scanner(System.in
        int n = sn.nextInt();
        boolean ans = isPrime(n);
        System.out.println(ans);
    }

    static boolean isPrime(int n){
        if(n<=1){
            return false;
        }
        int c=2;
        while(c*c<=n){
            if(n%c==0){
                return false;
            }
            c++;
        }
        return c*c>n;
    }
```

* Amstrong number

a = 153

$1^3 + 5^3 + 3^3$

$= 1 + 125 + 27$

$= 153$

Amstrong number

```
public class question {
    psvm (string [] args) {
    scanner sn = new scanner (system.in);
    int n = sn.nextInt();

    system.out.println (isArmstrong(n));
}

static boolean isArmstrong (int n) {
    int original = n;
    int sum = 0;

    While (n > 0) {
        int rem = n % 10;
        n = n / 10;
        sum = sum + rem * rem * rem;
    }
    return sum == original;
}
```