```
floating point data types
+++++++++++++++++++++++++
 To work with real numbers we use  the following datatypes
      a. float
      b. double

All the above datatypes are used to store whole number, if we want to store real
number
then we need to prefer using floating data type.

float
   If we want to store the data in terms of accuracy upto 5 decimal places then we
use
   float datatype.
   Corresponding wrapper class is java.lang.Float
   size:: 4bytes
   minrange::1.4E-45(MIN_VALUE)
   maxrange::3.4028235E38(MAX_VALUE)

eg:: float f=35.5;(double)
     float f=35.5f;(float)
     float f=35.5F;(float)
    To specify the float type we can prefix it with 'f' or 'F'.

double
   If we want to store the data in terms of accuracy upto 15 decimal places then we
use
   double datatype.
   Corresponding wrapper class is java.lang.Double
   size:: 8bytes
   minrange::4.9E-324(MIN_VALUE)
   maxrange::1.7976931348623157E308(MAX_VALUE)

eg:: double d=35.5;(valid)
     dobule d=35.5D;(valid)
     dobule d=35.5d;(valid)
    To specify the double type we can prefix it with 'd' or 'D'.


char types
==========
   It is represented by a single character with in a single quotes.
   eg:: char c='a';//valid
        char d=97;//valid
        char c='ab';//invalid
        char c= "a";//invalid
   Corresponding wrapper class is java.lang.Character
   size:: 2bytes(Since it supports Internationalization)

boolean types
=============
  The only allowed values for boolean type is true,false(case is also important)
  eg:: boolean b=false;
       boolean b=true;
       boolean b ="true";//CE:Incompatible types
       boolean b= 0;//CE:Incompatible types


Note: Reserve words for datatypes(byte,short,int,long,float,double,char,boolean)
```

byte(1)
short(2)
int(4)
long(8)
float(4)
double(8)
char(2)
boolean

Primitive Type casting
======================

a. The process of converting data from one type to another type is called TypeCasting

```
    There are 2 types
      1. Implicit type casting
      2. Explicit type casting
```

Implicit TypeCasting
====================

=> The process of converting data from lower data type to higher datatype is called "Implicit type casting".

```
 1       2        4      8        4        8
byte => short => int => long => float => double
                  ^
                  |
                 char
                  2
```

```
eg1:: byte b= 10;
      int i=b;
      System.out.println(b+" "+i);// 10 10

eg2:: int i=10;
      byte b= i;//CE:: loss of precession
      System.out.println(b+" "+i);

eg3:: byte b= 65;
      char c= b;//CE: loss of precession
      System.out.println(b+" "+c);

eg4:: char c ='A';
      short s=c;//CE: loss of precession
      System.out.println(c+" "+s);
```

Note:: byte and short internal representation is not compatible to convert into char.

```
eg5:: char c='A;
      int i=c;
      System.out.println(c+" "+i);// A 65
```

Note:: char internal representation is compatible with int type.

```
eg6::  byte b= 128;
       System.out.println(b);//CE: possible of loss or precession
```

if the value assigned to a variable,if it reaches the max limit then compiler will automatically type promote the data to next higher data type. In case of byte and short the next higher data type is "int".

```
eg7::  byte b1=4;
       byte b2=5;
       byte b3= b1 + b2;//CE:: byte + byte = int
       System.out.println(b3);

       byte b1=60;
       byte b2=70;
       byte b3= b1 + b2;//CE:: byte + byte = int
       System.out.println(b3);
```

```
Formulae:: In arithmetic operations the result will be always be
            Z= X + Y
            if X and Y belongs to {byte,short,int} then Z should be int.
            if either X or Y or both X and Y belongs to {long,float,double} then Z
is max(X,Y)
```

```
eg8:: long l= 10;
      float f=l;
      System.out.println(l + " " + f);//10 10.0

eg9:: float f=10.5f;
      long l= f;//CE:possible loss of precession
      System.out.println(l + " " + f);
```

Note: refer diagram to understand how long value can sit in float and float value can't sit in long.

Explicit TypeCasting

=> The process of converting data from higher data type to lower data type is called "Explicit Type casting".
=> JVM will do Explicit Type casting only on the instructions given by the programmer.
=> In case of Explicit Type casting there would be loss of data.

```
  Syntax:: P a  = (Q) b;
           P and Q should be primitive type and from Q to P there should be
implicit relationship.
```

```
eg::
   int i = 10;
   byte b=(byte)i;
   System.out.println(i);//10
   System.out.println(b);//10

eg2:: int i=10;
      short s= (byte)i;
      System.out.println(i);//10
      System.out.println(s);//10

eg3:: byte b= 65;
      char c= (char)b;
      System.out.println(b);//65
      System.out.println(c);//A
```

```
eg4:: char c= 'A';
      short s= (short)c;
      System.out.println(c);//A
      System.out.println(s);//65

eg5:: short s= 65;
      char c = (byte) s;//CE
      System.out.println(c);
      System.out.println(s);

eg6:: byte b1=10;
      byte b2=30;
      byte b3=(byte) b1 + b2;//CE
      System.out.println(b3);

eg7:: byte b1=10;
      byte b2=30;
      byte b3=(byte) (b1 + b2);
      System.out.println(b3);//40

eg8:: double d= 22.222;
      byte b= (byte)(long)(int)(short)d;
      System.out.println(d);//22.222
      System.out.println(b);//22

eg9:: int i = 130;
      byte b= (byte)i;
      System.out.println(b);//-126

      solution:: minrange + (result -maxrange-1)
```

```
UTF-8
   'A' => 65
   'a' => 97
    0  => 48
```

Note: Compiler -> Will performing typechecking(check value can be stored based on the range of values)
JVM -> Will allocate memory based on the datat type and performs the necessary operation(type casting)

++++++++
Snippets
++++++++

For the code below, what should be the name of java file?

1.

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

A. Hello.java
B. World.java
C. HelloWorld.java[Answer]
D. helloworld.java

Convention in java-> Which ever class contains main(),check that classname and save the file with classname.

2.

Does below code compile successfully?

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

A. yes[Answer]

B. no

3.

What is the signature of special main method?

A. public static void main(String args)

B. public static void main(String[] a)[Answer]

C. public static void main()

D. private static void main(String[] args)

4. What will be the result of compiling and executing Test class?

```
public class Test {
    public static void main(String[] args){
        byte b1 = ( byte ) ( 127 + 21 );
        System.out.println(b1);//-108
    }
}
```

A. 148

B. Compilation Error

C. -108[Answer]

D. -128

```
Output: minRange + (result -maxRange-1)
        -128     + (148 -127-1)
        -128     + 20
        -108
```

5.

Consider below code:

```
public class Test {
    public static void main(String[] args) {
        char c = 'Z';
        long l = 100_00l;
        int i = 9_2;
        float f = 2.02f;
        double d = 10_0.35d;

        l = c + i; // char + int  = int --> long
        f = c * l * i * f; // char * long * int * float --> float
        f = l + i + c; // long + int + char ----> long ---> float
        i = (int)d; //possible
        f = (long)d;//long -> float
    }
}
```

Does above code compile successfully?

A. Yes[Answer]

B. No

6.

Consider below code of Test.java file:

```
public class Test {
    public static void main(String[] args) {
        char c1 = 'a'; //UNICODE VALUE  'a' is 97
        int i1 = c1; //Line n1
        System.out.println(i1); //Line n2
    }
}
```

What is the result of compiling and executing Test class?

A. a
B. 97[Answer]
C. Line n1 causes compilation failure
D. Line n2 causes runtime error.