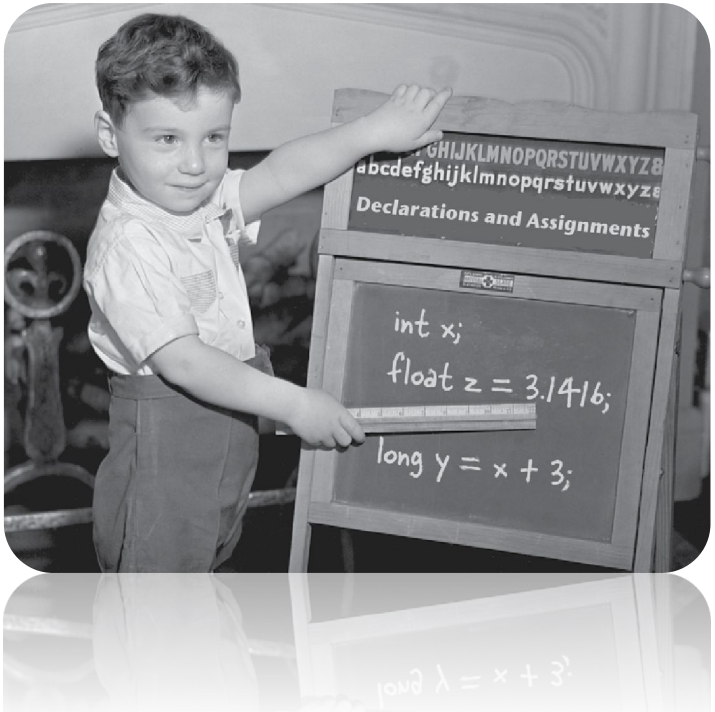
Explore | Expand | Enrich

SAMPLE PPT
IN
JAVA

Java Variables & Data Types



JAVA VARIABLES

Variables are containers for storing data values.\

- In Java, there are different types of variables,

  - ✔ **String -** stores text, such as "Hello". String values are surrounded by double quotes
  - ✔ **int -** stores integers (whole numbers), without decimals, such as 123 or -123
  - ✔ **float -** stores floating point numbers, with decimals, such as 19.99 or -19.99
  - ✔ **char -** stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
  - ✔ **boolean -** stores values with two states: true or false

DECLARING (CREATING) VARIABLES

- ✔ **Variables must have a type**
- ✔ Variables must have a **name**

```
int count;
```

type — name

DECLARING (CREATING) VARIABLES

To create a variable, you must specify the type and assign it a value:

Syntax:

```
type variable = value;
```

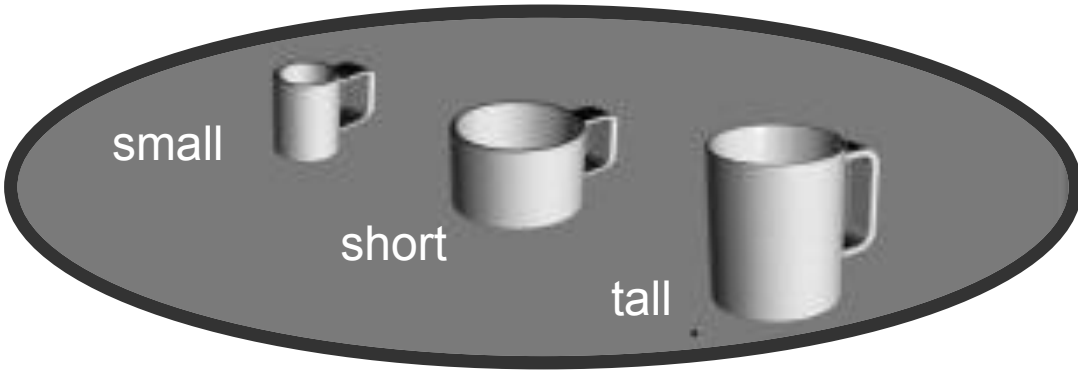
Example: 01

```
String name = "John";
System.out.println(name);
```

```
public class MyClass {
  public static void main(String[] args) {
    String name = "John";
    System.out.println(name);
  }
}
```

DECLARING (CREATING) VARIABLES

A variable is just a cup. A container. It *holds* something.



DECLARING (CREATING) VARIABLES

Syntax:

```
int myNum = 5;
float myFloatNum = 5.99f;
char myLetter = 'D';
boolean myBool = true;
String myText = "Hello";
```

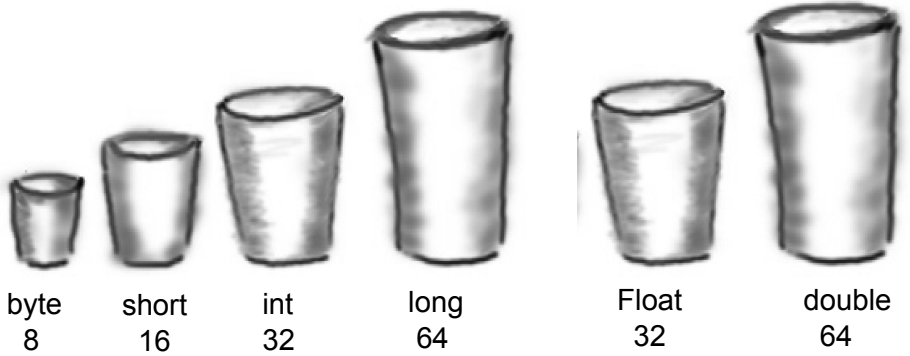
Example: 02

```
public class MyClass {
  public static void main(String[] args) {
    int myNum = 15;
    System.out.println(myNum);
  }
}
```

DECLARING (CREATING) VARIABLES

Example: 03

```
int x;
x = 234;
byte b = 89;
boolean isFun = true;
double d = 3456.98;
char c = 'f';
int z = x;
boolean isPunkRock;
isPunkRock = false;
boolean powerOn;
powerOn = isFun;
long big = 3456789;
float f = 32.5f;
```



Note the 'f'. glass have that with a float, because Java thinks anything with a floating point is a double, unless you use 'f'.

DISPLAY VARIABLES

The println() method is often used to display variables.

- To combine both text and a variable, use the + character:

Example: 03

```
String name = "John";
System.out.println("Hello " + name);
```

```
public class MyClass {
  public static void main(String[] args){
    String name = "John";
    System.out.println("Hello " + name);
  }
}
```

DISPLAY VARIABLES

Example: 04

```
String firstName = "John ";
String lastName = "Doe";
String fullName = firstName + lastName;
System.out.println(fullName);
```

Example: 05

```
int x = 5;
int y = 6;
System.out.println(x + y);
```

JAVA DATA TYPES

Variable in Java must be a specified data type

Example:

```
int myNum = 5;
float myFloatNum = 5.99f;
char myLetter = 'D';
boolean myBool = true;
String myText = "Hello";
```

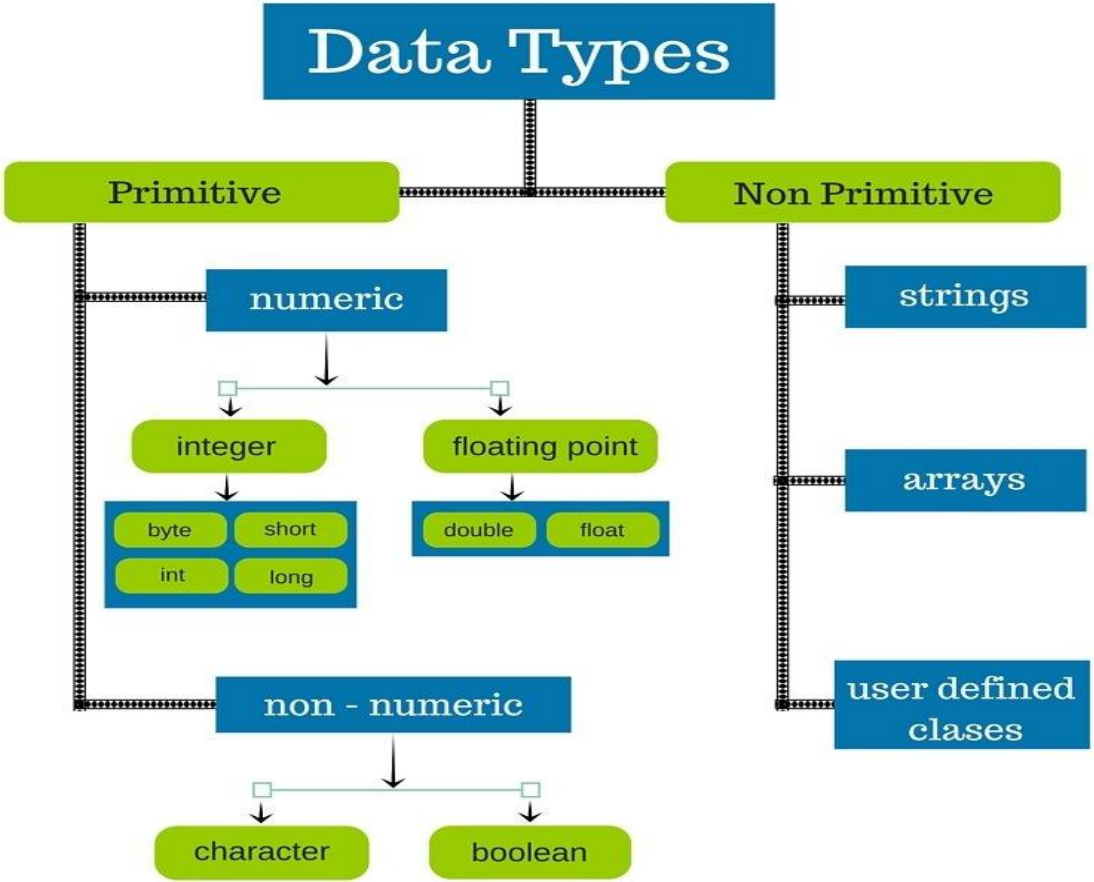
```
public class MyClass {
  public static void main(String[] args) {
    int myNum = 5;
    float myFloatNum = 5.99f;
    char myLetter = 'D';
    boolean myBool = true;
    String myText = "Hello";
    System.out.println(myNum);
    System.out.println(myFloatNum);
    System.out.println(myLetter);
    System.out.println(myBool);
    System.out.println(myText);
  }
}
```

JAVA IDENTIFIERS

- All Java **variables** must be **identified** with **unique names**
- These unique names are called **identifiers**
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume)

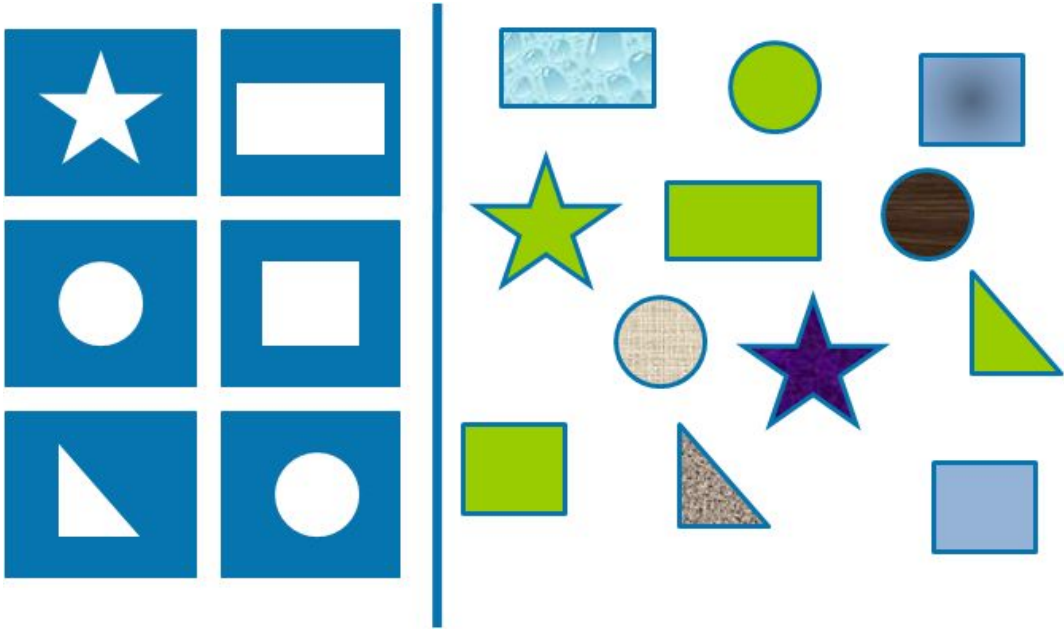
THE GENERAL RULES FOR CONSTRUCTING NAMES FOR VARIABLES

- Names can contain letters, digits, underscores, and dollar signs
- Names should begin with a letter
- Names can also begin with $
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Names should start with a lowercase or uppercase letter and it cannot contain whitespace
- Reserved words (like Java keywords, such as int or String) cannot be used as names

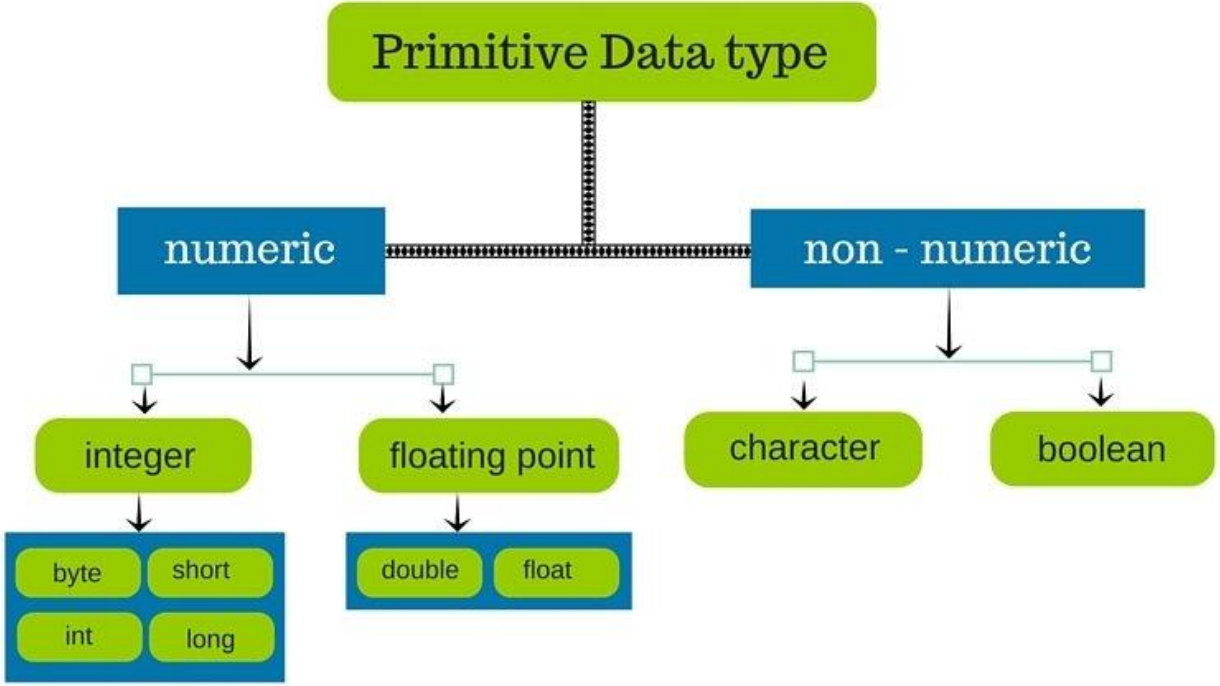

Data Types
Primitive
Non Primitive
numeric
integer
floating point
byte
short
int
long
double
float
non - numeric
character
boolean
strings
arrays
user defined clases

PRIMITIVE DATA TYPES

- ✔ A primitive data type specifies the size and type of variable values.
- ✔ Primitive Data Type are inherently supported by the programming language.
- ✔ Primitive Data Type are also called as predefined data types.



PRIMITIVE DATA TYPES



PRIMITIVE DATA TYPES

Data Type	Size	Range Inclusive
byte	1 byte	-128 to 127
short	2 bytes	-32,768 to 32,767
int	4 bytes	-2,147,483,648 to 2,147,483,647
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	3.4e−038 to 3.4e+038. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	1.7e−308 to 1.7e+038. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

PRIMITIVE DATA TYPES

- Depending upon the requirement we should choose the appropriate data type
- The size allowed is calculated using the number of bytes

Example:

for byte the size is 8-bits, so the range allowed is 2 power 8 which is 256. This 256 is split into half to support both negative and positive numbers.
That is reason the size is -128 to +127. The positive side is +127 instead of +128, because zero is also included in the range

0	0
0	1
1	0
1	1

2 bits
2^2 = 4

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

3 bits
2^3 = 8

NUMBERS

Primitive number types are divided into two groups:

1. Integer types stores whole numbers, positive or negative (such as 123 or -456), without decimals,

- types : byte, short, int and long

2. Floating point types represents numbers with a fractional part,

- types: float and double

INTEGER TYPES

Byte

- The byte data type can store whole numbers from -128 to 127
- Example

```
byte myNum = 100;
System.out.println(myNum);
```

```
public class MyClass {
public static void main(String[] args){
  byte myNum = 100;
  System.out.println(myNum);
  }
}
```

INTEGER TYPES

Short

- The short data type can store whole numbers from -32768 to 32767
- Example

```
short myNum = 5000;
System.out.println(myNum);
```

```
public class MyClass {
  public static void main(String[] args) {
    short myNum = 5000;
    System.out.println(myNum);
  }
}
```

INTEGER TYPES

Int

- The int data type can store whole numbers from -2147483648 to 2147483647
- Example

```
int myNum = 100000;
System.out.println(myNum);
```

```
public class MyClass {
  public static void main(String[] args) {
    int myNum = 100000;
    System.out.println(myNum);
  }
}
```

INTEGER TYPES

Long

- The long data type can store whole numbers from -9223372036854775808 to 9223372036854775807
- Example

```
long myNum = 15000000000L;
System.out.println(myNum);
```

```
public class MyClass {
  public static void main(String[] args) {
    long myNum = 15000000000L;
    System.out.println(myNum);
  }
}
```

FLOATING POINT TYPES

Float

- The float data type can store fractional numbers from 3.4e−038 to 3.4e+038
- Example

```
float myNum = 5.75f;
System.out.println(myNum);
```

```
public class MyClass {
  public static void main(String[] args) {
    float myNum = 5.75f;
    System.out.println(myNum);
  }
}
```

FLOATING POINT TYPES

Double

- The double data type can store fractional numbers from 1.7e−308 to 1.7e+038
- Example

```
double myNum = 19.99d;
System.out.println(myNum);
```

```
public class MyClass {
  public static void main(String[] args) {
    double myNum = 19.99d;
    System.out.println(myNum);
  }
}
```

FLOATING POINT TYPES

Booleans

- A boolean data type is declared with the boolean keyword and can only take the values true or false
- Example

```
boolean isJavaFun = true;
boolean isFishTasty = false;
System.out.println(isJavaFun);

System.out.println(isFishTasty);
```

```
public class MyClass {
  public static void main(String[] args) {
    boolean isJavaFun = true;
    boolean isFishTasty = false;
    System.out.println(isJavaFun);
    System.out.println(isFishTasty);
  }
}
```

FLOATING POINT TYPES

Characters

- The char data type is used to store a single character
- The character must be surrounded by single quotes, like 'A' or 'c'
- Example

```
char myGrade = 'B';
System.out.println(myGrade);
```

```
public class MyClass {
  public static void main(String[] args) {
    char myGrade = 'B';
    System.out.println(myGrade);
  }
}
```

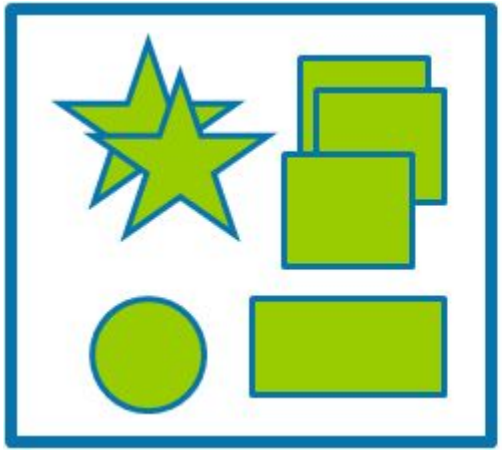
EXAMPLE


```
char a = 65, b = 66, c = 67;
System.out.println(a);
System.out.println(b);
System.out.println(c);
```

```
public class MyClass {
  public static void main(String[] args) {
    char a = 65, b = 66, c = 67;
    System.out.println(a);
    System.out.println(b);
    System.out.println(c);
  }
}
```

NON-PRIMITIVE DATA TYPES

Non-primitive data types are called **reference types** because they refer to objects



FLOATING POINT TYPES

Strings

- The String data type is used to store a sequence of characters (text)
- String values must be surrounded by double quotes
- Example

```
String greeting = "Hello World";
System.out.println(greeting);
```

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

NON-PRIMITIVE DATA TYPES

Non-primitive data types are called **reference types** because they refer to objects

• Primitive types are predefined in Java. • primitive types cannot be used to call methods to perform certain operations, • A primitive type has always a value, • A primitive type starts with a lowercase letter, • The size of a primitive type depends on the data type,	• Non-primitive types are created by the programmer and is not defined by Java (except for String). • Non-primitive types can be used to call methods to perform certain operations, • non-primitive types can be null. • non-primitive types starts with an uppercase letter. • non-primitive types have all the same size.

EXERCISE:

Add the correct data type for the following variables

```
_____ myNum = 9;
_____ myFloatNum = 8.99f;
_____ myLetter = 'A';
_____ myBool = false;
_____ myText = "Hello World";
```

JAVA TYPE CASTING

You really don't want to spill that...

Be sure the value can fit into the variable.

Example:

```
int x = 24;
byte b = x;
//won't work!!
```

JAVA TYPE CASTING

Type casting is when you assign a value of one primitive data type to another type

- In Java, there are two types of casting
  - **Widening Casting (automatically) -** converting a smaller type to a larger type size

    byte -> short -> char -> int -> long -> float -> double
  - **Narrowing Casting (manually) -** converting a larger type to a smaller size type

    double -> float -> long -> int -> char -> short -> byte

WIDENING CASTING

Widening casting is done automatically when passing a smaller size type to a larger size type

Example:

```
public class MyClass {
  public static void main(String[] args) {
    double myDouble = 9.78;
    int myInt = (int) myDouble;
    System.out.println(myDouble);
    System.out.println(myInt);
  }
}
```

CONSIDER THIS CODE SNIPPET

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.print("Y" + "O");
        System.out.print('L' + 'O');
    }
}
```

Can you predict the output?

YOLO 

YO155 

CONSIDER THIS CODE SNIPPET

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.print("Y" + "O");
        System.out.print('L');
        System.out.print('O');
    }
}
```

Can you predict the output?

YO7679 

YOLO 

NARROWING CASTING

Narrowing casting must be done manually by placing the type in parantheses in front of the value

Example:

```
public class MyClass {
  public static void main(String[] args) {
    int myInt = 9;
    double myDouble =myInt;
    System.out.println(myInt);
    System.out.println(myDouble);
  }
}
```

GUESS THE OUTPUT

```
public class Test
{
  public static void main(String[] argv)
  {
    char ch = 'c';
    int num = 88;
    ch = num;
  }
}
```

Answer: **Error**

GUESS THE OUTPUT

```
class Simple
{
     public static void main(String[] args)
     {
     float f=10.5f;
     //int a=f;//Compile time error
     int a=(int)f;
     System.out.println(f);
     System.out.println(a);
     }
}
```

A.10.5 ✓

B.10

PREDICT THE OUTPUT

```
class booloperators{
public static void main(String args[])
{
     boolean var1 = true;
     boolean var2 = false;
System.out.println((var1 & var2));
     }
}
```

A. 0
B. 1
C. true
D. false ✓

PREDICT THE OUTPUT

```
class asciicodes
{
  public static void main(String args[])
   {
        char var1 = 'A';
        char var2 = 'a';
        System.out.println((int)var1 + " " + (int)var2);
    }
}
```

A. 162

B. 65 97 ✓

C. 67 95

D. 66 98

PREDICT THE OUTPUT

```
class A{
   public static void main(String args[]){
                byte b;
            int i = 258;
            double d = 325.59;
            b = (byte) i;
            System.out.print(b);
            i = (int) d;
            System.out.print(i);
                b = (byte) d;
                System.out.print(b);
        } }
```

A. 258 325 325
B. 258 326 326 ✓
C. 2 325 69
D. Error

PREDICT THE OUTPUT

```
class A
{
  public static void main(String args[])
    {
      int x;
      x = 10;
      if(x == 10)
       {
         int y = 20;
         System.out.print("x and y: "+ x + " " + y);
         y = x*2;
      }
        y = 100;
        System.out.print("x and y: " + x + " " + y);
    }}
```

A. 10 20 10 100

B. 10 20 10 20

C. 10 20 10 10

D. Error ✓

PREDICT THE OUTPUT

```
public class Test
 {
    static void test(float x)
     {
          System.out.print("float");
   }
   static void test(double x)
     {
           System.out.print("double");
     }
   public static void main(String[] args)
     {
           test(99.9);
     }
}
```

A. float

B. double ✓

C. Compilation Error

D. Exception is thrown at runtime

PREDICT THE OUTPUT

```
public class Test
{
  public static void main(String[] args)
  {
      int i = 010;
      int j = 07;
      System.out.println(i);
      System.out.println(j);
   }
 }
```

A. 8 7 ✓

B. 10 7

C. Compilation fails with an error at line 3

D. Compilation fails with an error at line 5

E. None of these

PREDICT THE OUTPUT

```
public class MyClass
{
     public static void main(String[] args)
     {
          int a = 10;
          System.out.println(++a++);
     }
}
```

A. 10

B. 11

C. 12

D. Compilation Error ✓

PREDICT THE OUTPUT

```
public class Main
{
    public static void main(String[] args)
    {
        int a = 5+5*2+2*2+(2*3);
        System.out.println(a);
    }
}
```

A. 138

B. 264

C. 41

D. 25 ✓

PREDICT THE OUTPUT

```
class char_increment {
public static void main(String args[])
{
      char c1 = 'D';
      char c2 = 84;
      c2++;
      c1++;
      System.out.println(c1 + " " + c2);
   }
}
```

A. E U ✓

B. U E

C. V E

D. U F

PREDICT THE OUTPUT

```
class conversion
{
 public static void main(String args[])
 {
     double a = 295.04;
     int b = 300;
     byte c = (byte) a;
     byte d = (byte) b;
     System.out.println(c + " " + d);
 }
}
```

A. 38 43

B. 39 44 ✓

C. 295 300

D. 295.04 300

PREDICT THE OUTPUT

```
class mainclass
 {
   public static void main(String args[]){
    char a = 'A';
    a++;
    System.out.print((int)a);
 }
}
```

A. 66 ✓

B. 67

C. 65

D. 64

PREDICT THE OUTPUT

```
class variable_scope
{
  public static void main(String args[]){
     int x;
     x = 5;
      {
    int y = 6;
    System.out.print(x + " " + y);
      }
    System.out.println(x + " " + y);
   }
}
```

A. 5 6 5 6

B. 5 6 5

C. Runtime error

D. Compilation error ✓

OUR PRODUCT

Explore | Expand | Enrich

Aptimithra

aptimithra.com

Google Review

ethnus.com


THANK YOU