Java - Day1

Basics of java

Data Types and Operators

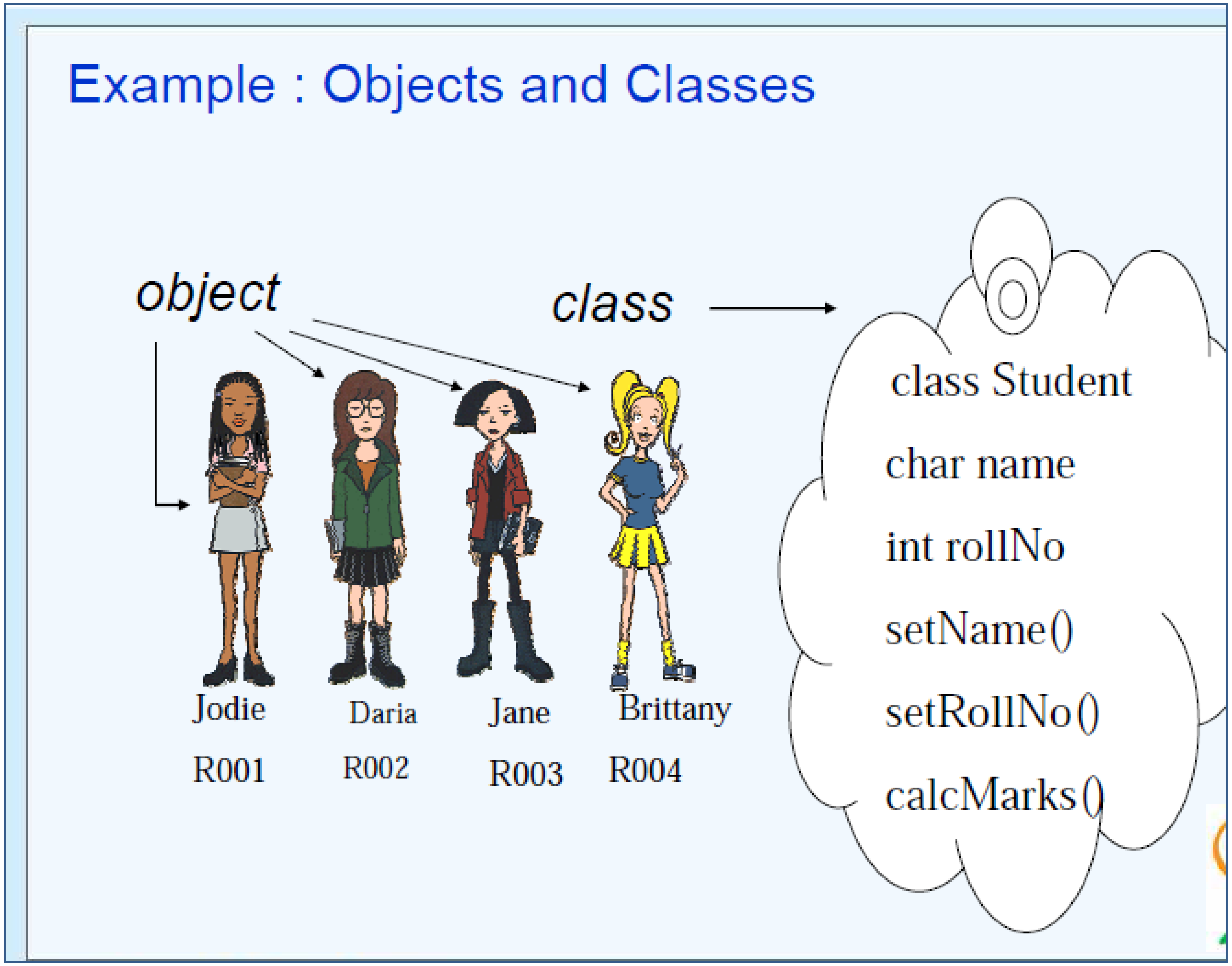
Control Structures

Class and Objects

- **Class:** A class is a blueprint or prototype of a real world entity.
- It defines variables and methods(functions) common to all objects of that class.

- **Object:** It is a specimen or representative of a class.
- It models real world objects you find in everyday life.

Example: Objects and Classes



Features of OOP

- **Abstraction** : Extracting the essential information and hiding irrelevant details.
- **Encapsulation** : The process of binding code and data together in the form of a capsule.
- **Inheritance**: The feature by which a class acquires properties and functionalities of another class.
- **Polymorphism**: The feature that allows the same interface to be used for a general set of action.

Features of Java

- Object Oriented
- Simple - Compared to earlier languages like C++
- Robust- Errors are corrected while writing code , well before run time
- Architecture Neutral/ Portable – Java code complied on Windows machine can be run on other OS without recompilation
- Secure
- Multithreading

Java Virtual Machine (JVM) (1 of 2)

- Source code is stored in a .java file.
- Java Compiler compiles .java file into .class file which is a bytecode.
- Bytecode is interpreted by JVM
- JVM is like a processor(virtual machine) implemented with software.

JVM (2 of 2)

- Interface of a JVM to a .class file is same , irrespective of underlying OS- This makes platform independence easier

Hello World!!

- public class FirstProgram {
- public static void main(String[] args)
- {
- System.out.println(“Hello World!!”);
- }
- }
- // save as FirstProgram.java
- // this is source code

Compilation and Execution

Step 1: Java Program(.java)

Step 2: Java Compiler (javac)

Step 3: ByteCode (.class)

Step 4:

Interpreter	Interpreter	Interpreter
Windows	Linux	Mac

Best Practices

- One .java file should have one public class and other default classes , if any.
- Name of file must be same as name of class.
- Stand alone java program must have public static void main defined.

 - it is the starting point of program

- not all classes need main method

- Must follow indentation and coding standards

Data Types in Java

- Java is a strongly typed language. It means every declared variable must have a declared type.
- Two Types:

- Primitive Types
- Reference Types

Primitive Data types

•

Data Types	Type	Storage Requirement
Integer	byte	1 byte
	short	2 bytes
	int	4 bytes
	long	8 bytes
Floating point	Float	4 bytes
	double	8 bytes
Textual	char	2 bytes
Logical	boolean	1 byte (true/false)

Reference Types

- A reference variable is required to access an object of a class.
- We create an object of a class using

  new classname()

This object is accessed by reference variable.

For eg.,

Animal a=new animal();

a.eat(); // a is reference variable, which refers to animal object and accesses eat() method.

Comments in java

- Single Line comments:

```
// this is a comment
```

- Multiline Comments:

```
/* this is a
Multiline
Comment  */
```

Variables

- Variables must have a data type:

```
int count;
int max=100;
```

- Variables can be declared anywhere in program. Declare it as and when required.
- If a variable is used without initializing it, the compiler will show an error.

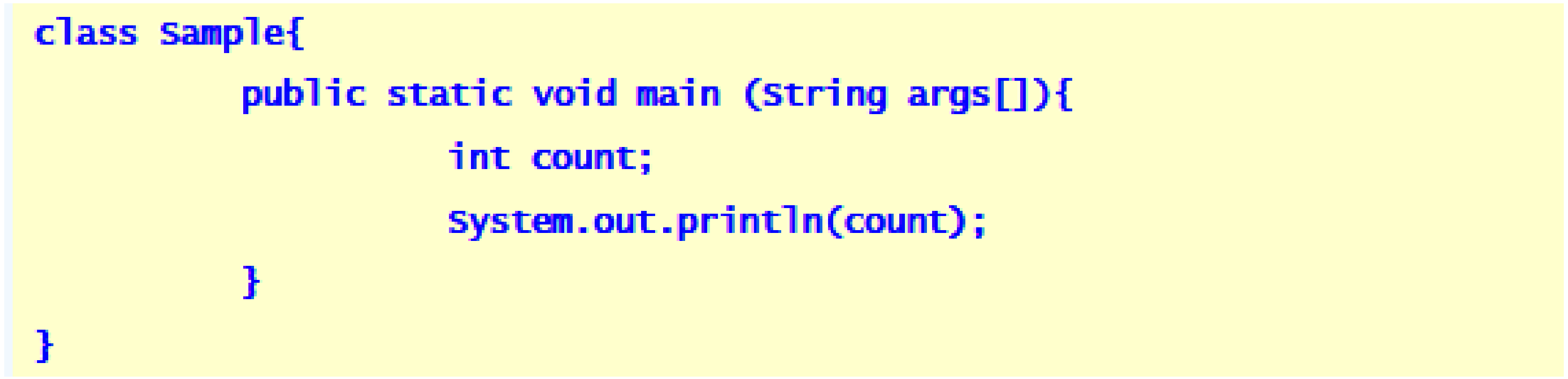
Variable declaration and assignment

```
int count;   // declaration
count=10   // assignment or initialization
```

```
int num=20; // declaration and assignment in
    same step
```

*** variable must have a type and a name.

What is the output of below program?....



```
class Sample{
        public static void main (String args[]){
                int count;
                System.out.println(count);
        }
}
```

Typecasting of primitive data types - 1

- Automatic or implicit type conversion:

  - variable of smaller capacity can be assigned to variable of bigger capacity

  int i=10;

  double d;

  d=I;

Typecasting of primitive data types - 2

- Explicit Type conversion
- Variable of bigger capacity is assigned to a variable of smaller capacity with a probable loss of data using type cast operator:

```
    double d=10;
   int i;
 i=(int )d;
```

Access Modifiers: Private and Public

- Data members are always kept Private.

  - It is accessible only within the class

- The method which expose the behaviour of object are kept public.

  - We can have other helper methods which are private

- Key features of object oriented programs

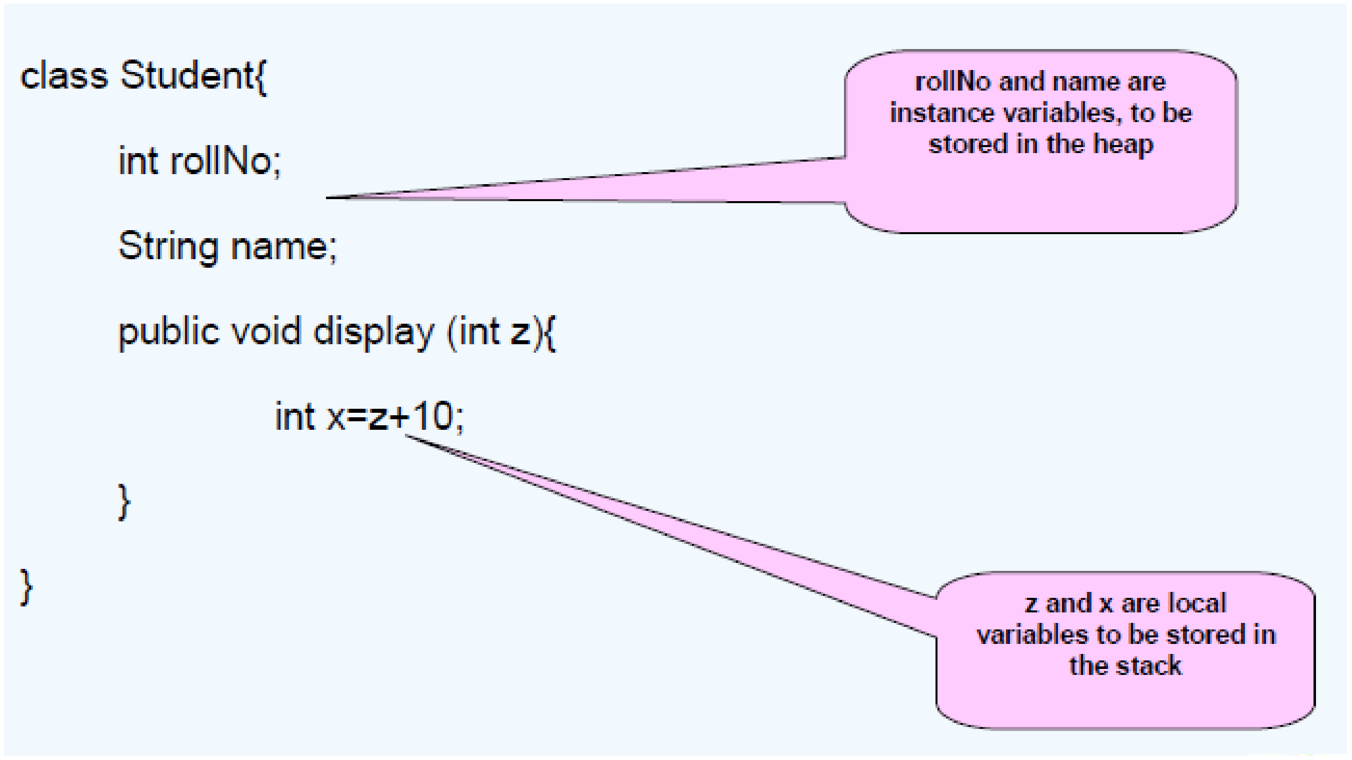
  - encapsulation: Binding of code and data together.

  -State is hidden and behaviour is exposed to external world.

Variables and their scope ..1

- Instance Variables (Member variables)
  - declared inside a class
  -outside any method or constructor
  -Lifetime depends on lifetime of object
- Local Variables
  - declared inside a method
  - method parameters are also local variables
  -lifetime ends when program call ends

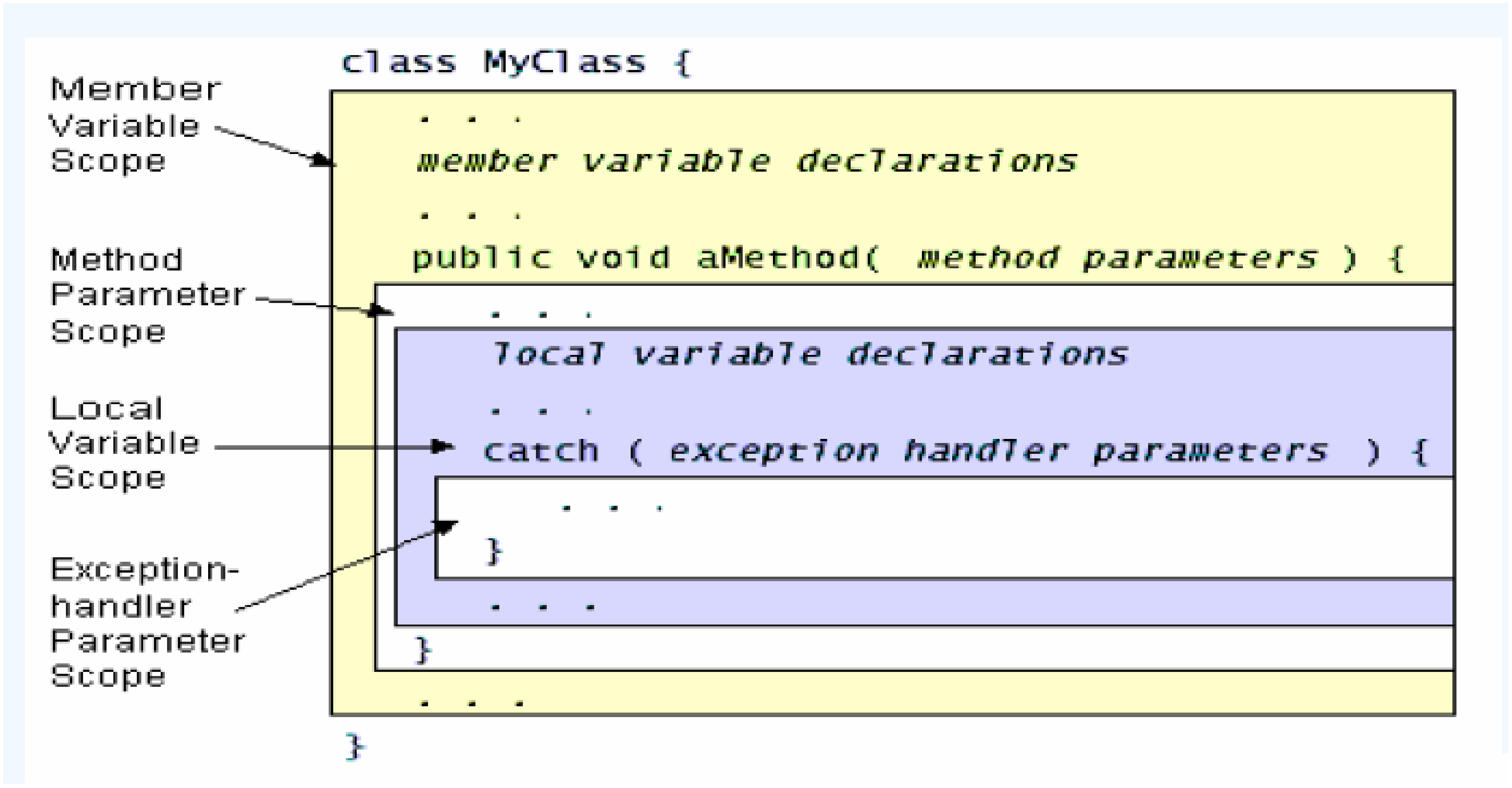
Variables and their scope ..2



```
class Student{
        int rollNo;
        String name;
        public void display (int z){
                int x=z+10;
        }
}
```

rollNo and name are instance variables, to be stored in the heap
z and x are local variables to be stored in the stack

Variables and their scope ..3



```
class MyClass {
    . . .
    member variable declarations
    . . .
    public void aMethod( method parameters ) {
        . . .
        local variable declarations
        . . .
        catch ( exception handler parameters ) {
            . . .
        }
        . . .
    }
    . . .
}
```

Operators

- Arithmetic Operators: + - * /
- + addition
- - subtraction
- * multiplication
- / integer division if both operands are integers else floating point division
- Integer Remainder is denoted by % (modulus).
- 17/2 is 8, 17%2 is 1, 17.0/2 is 8.5

Increment and Decrement Operators

- ++ adds 1 to the current value of variable
- -- subtracts 1 from the current value of variable.
- Postfix and prefix notations:

```
int  m=7;
int  n=7;
int  a=2*++m; //a=16, m=8
int  b=2*n++; //b=14,n=8
```

Relational and Boolean Operators

- Equality: ==
- Inequality: !=
- Less than <
- Less than or equals to <=
- Greater than >
- Greater than or equals to >=
- Logical And - &&
- Logical Or - ||

Arrays in Java ..1

- It is a data structure with ordered collection of a fixed number of homogeneous data elements.
- Size of an array is fixed.
- Array can be of primitive data types of reference variable type.
- All elements in an array must be of same data type.

Arrays in Java ..2

- Declaring array variables

        <element type> [] <array name>;
                Or
        <element type> <array name>[];

- For example:

  int intArray[];

  Animal[] landAnimals, waterAnimals ;

Arrays in Java ..2

- Constructing an Array

  <array name>= new <elementType>[<No of elements>];

For Example:

```
   int intArray[];
intArray=new int[10];
```

- Combined Declaration and construction:

  ```
  int intArray[]=new int[10];
  ```

Arrays in Java ..3

- Declaring and Initializing an array

  int intArray[]={1,2,3,4,5};

  char charArray[]= { ‘a’, ‘s’, ‘ d’ };

  Animal animalArray[]= {new Animal(), new Animal () };

Arrays in Java ..4

- Java does not allow to extend its boundaries.
- If x is reference to an array, then

  x.length

  will give length of array

Control Structures in Java

- Conditional Structures:
  - if Statement
  - if else
  - if Elseif
  - if Elseif Else
  - Switch Case

- Looping Structures:

- For loop:

```
for( initialization; condition; increment)
      { // statements
      }

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

Exercise:

- Write a program to print even numbers between 0 to 25 considering 0 as even number.
- Write a program to print average of int array:

  int intArray[]={5,10,15,20,25,30}
- Write a program to print fibonacci series(0 1 1 2 3 5 8 13 21 34 …).

While

- While Loop: It executes a statement (or a block) while a condition is true.

```
        While (condition) statement;
```

Loop will never execute if condition is false.

Do - while

- If you wish to make sure that block executes at least once, then use do-while loop.

```
   do
   {
   // statements
   }
while (condition);
```

Multiple Selection: Switch Statement

- Execution starts at case label that matches with the input/selected value until the next break or end of switch.
- If none of case matches then default is executed , if present.

- Scanner in=new Scanner(system.in);
- System.out.println (“select option 1, 2 or 3”);
- int choice=in.nextInt();
- switch(choice)
- {
- case 1:
- …
- break;
- case 2:
- …
- break;
- ….
- default:
- …
- break;
- }