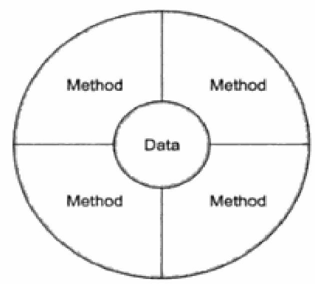
B.Sc (Computer Science)
Object Oriented Programming with Java and Data Structures
Unit-I

1. Define OOPS. Explain the main concepts of Object Oriented Programming. (Mar 2010)

Define (a) Data Abstraction. (b) Polymorphism. (Oct 2012)

OOPS stands for Object Oriented Programming System. It is the most popular approach followed by programming languages. Java is one such language which follows OOPS.



OOP treat data as a critical element in the programming development and does not allow it to flow freely around the system. It ties data more closely to the functions. OOP allows decomposition of a problem into a number of entities called objects and build data and functions around these objects. The data of an object can be accessed only by the functions of one object can access the functions of other objects. OOP allows us to decompose a problem into a number of entities called "Objects" and then build data and methods.

Some of the important features of Object-oriented programming are:

- Emphasis is on data rather than procedure.
- Programs are divided into what are known as objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together in the data structure.
- Data is hidden and cannot be accessed by external functions.
- Objects may communicate with each other through functions.
- Follows bottom-up approach in program design.

It supports the following concepts:

Objects: Objects are the basic runtime entities in an object-oriented system. They may represent a person, a place, a bank account or any thing that the program can handle.

Classes: A class is a collection of objects of similar type. The entire set of data and code of an object can be made a user-defined data type using the concept of a class. A class may be thought of as a data type and an object as a variable of that type. Once a class has been defined, we can create any number of objects belonging to that class.

Data Abstraction: Abstraction refers to the act of representing essential features without including the background details or explanation. Classes use the concept of abstraction such as size, weight and cost and the methods that operate on these attributes. It is a mechanism used to create new data types that include several related operations to be performed on it and attributes to suit the requirements of an application.

Data Encapsulation: Encapsulation means "combining". This feature allows combining several related properties (attributes) and functions (methods) into a single unit as its members. The wrapping up of data and methods into a single unit (class) is known as "Encapsulation". The members of that unit are accessible to its instances depending up on their accessibility conditions. The data is not accessible to the outside world and only those methods, which are wrapped in the class, can access it.

Inheritance: This feature is used to derive the properties of an existing object. Inheritance is the process by which objects of one class acquire the properties of objects of another class.

Polymorphism: It means "many forms". Different behavior of functions or operators or objects at different situations can be called as "polymorphism". Polymorphism can be exhibited in three different ways: Function Overloading, Operator overloading and Dynamic binding.

2. What is Message Communication? (Mar 2012)

An object-oriented program consists of a set of objects that communicate with each other. The process of programming in an object-oriented language, involves the following basic steps:

1. Creating classes that define objects and their behavior.
2. Creating objects from class definitions.
3. Establishing communication among objects.

Objects communicate with one another by sending and receiving information just the same way as people pass messages to one another. A message for an object is a request for execution of a procedure, and therefore will invoke a method in the receiving object that generates the desired results.

3. List the benefits of Java/OOP. (Mar 2010) (Oct 2010) (Oct 2011)

Benefits of OOP:

Object-Oriented Programming contributes to the solution of many problems associated with the development and quality of software products.

The advantages of OOP are:

- Through inheritance we can eliminate redundant code and extend the use of existing classes.
- We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch.
- Data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple objects to coexist without any interference.
- It is easy to map objects in the problem domain to those objects in the program.
- It is easy to partition the work in a project based on objects.
- The data-centered design approach enables us to capture details of a model in implementable form.

- Object-oriented systems can be easily upgraded from small to large systems.
- Message passing techniques for communication between objects make the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

4. Explain the features of Java. (Oct 2010) (Oct 2011) (Mar 2012) (Oct 2012)

Java has many advanced features when compared to other programming languages.

- Compiled and Interpreted
- Platform-Independent and Portable
- Object-Oriented
- Robust and Secure
- Distributed
- Familiar, Simple and Small
- Multithreaded and Interactive
- High Performance
- Dynamic and Extensible

Compiled and Interpreted: First, Java compiler translates source code into *bytecode* instructions. Bytecodes are not machine instructions and therefore, in the second stage, Java interpreter generates machine code that can be directly executed by the machine that is running the Java program. We can thus say that Java is both a compiled and an interpreted language.

Platform-Independent and Portable: Java programs can be easily moved from one computer system to another, *anywhere* and *anytime.* Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. This is the reason why Java has become a popular language for programming on Internet.

Object-Oriented: Java is a true object-oriented language. Almost everything in Java is an *object.* All program code and data reside within objects and classes. Java comes with an extensive set of *classes,* arranged in *packages,* that we can use in our programs by inheritance. The object model in Java is simple and easy to extend.

Robust and Secure: Java is a robust language. It provides many safeguards to ensure reliable code. It has strict compile time and run time checking for data types and errors. It provides automatic garbage-collection. Java also allows exception handling to catch runtime errors.

Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations without proper authorization.

Distributed: Java is designed as a distributed language for creating applications on networks. It has the ability to share both data and programs. Java applications can open and access remote objects on Internet. This enables multiple programmers at remote locations to collaborate and work together on a single project.

Simple, Small and Familiar: Java is a small and simple language. Familiarity is another important feature of Java. To make the language look familiar to the existing programmers, it was modeled on C and C++ languages. Java uses many constructs of C and C++ and therefore, Java code "looks like a C++" code. In fact, Java is a simplified version of C++.

Multithreaded and Interactive: Multithreaded means handling multiple tasks simultaneously. Java supports multithreaded programs; This means that we need not wait for the application to finish one task before beginning another. For example, we can listen to an audio clip while scrolling a page and at the same time download an applet from a distant computer.

High Performance: Java architecture is designed to reduce overheads during runtime. Further, the incorporation of multithreading enhances the overall execution speed of Java programs.

Dynamic and Extensible: Java is a dynamic language. Java is capable of dynamically linking in new class libraries, methods, and objects. Java programs support functions written in other languages such as C and C++. These functions are known as *native methods.* This facility enables the programmers to use the efficient functions available in these languages.

5. Explain the difference between C & Java. (Oct 2011)

Java is a lot like C but the major difference between Java and C is that Java is an object-oriented language and has mechanism to define classes and objects. In an effort to build a simple and safe; language, the Java team did not include some of the C features in Java.

- Java does not include the C unique statement keywords **sizeof,** and **typedef.**
- Java does not contain the data types **struct** and **union.**
- Java does not define the type modifiers keywords **auto, extern, register, signed,** and **unsigned,**
- Java does not support an explicit pointer type.
- Java does not have a preprocessor and therefore we cannot use # **define,** **# include,** and # **ifdef** statements.
- Java requires that the functions with no arguments must be declared with empty parenthesis and not with the v**oid** keyword as done in C.
- Java adds new operators such as **instanceof and >>>.**
- Java adds labelled **break** and **continue** statements.
- Java does not have “goto” control structure.
- Java adds many features required for object-oriented programming.

6. How Java differs from C++. (Mar 2010)

Java is a true object-oriented language while C++ is basically C with object-oriented extension. That is what exactly the increment operator + + indicates. C+ + has maintained backward compatibility with C. It is therefore possible to write an old style C program and run it successfully under C++. Java appears to be similar to C++ when we consider only the "extension" part of C++. However, some object-oriented features of C++ make the C++ code extremely difficult to follow and maintain.

- Java does not support operator overloading.
- Java does not have template classes as in C++.
- Java does not support multiple inheritances of classes. This is accomplished using a new feature called "interface".
- Java does not support global variables. Every variable and method is declared within a class and forms part of that class.
- Java does not use pointers.
- Java has replaced the destructor function with a finalize() function.
- There are no header files in Java.

7. Explain the contribution of Java to Internet. (Oct 2012)

Java and Internet: Java is strongly associated with the Internet. Internet users can use Java to create applet programs and run them locally using a "Java-enabled browser" such as HotJava. They can also use a Java-enabled browser to download an applet located on a computer anywhere in the Internet and run it on his local computer. In fact, Java applets have made the Internet a true extension of the storage system of the local computer.

Internet users can also setup their websites containing java applets that could be used by other remote users of Internet. This feature made Java most popular programming language for Internet.

8. What is WWW? What is the contribution of Java to WWW. (Mar 2011)

Java and World Wide Web: World Wide Web (WWW) is an open-ended information retrieval system designed to be used in the Internet's distributed environment. This system contains Web pages that provide both information and controls. Web system is open-ended and we can navigate to a new document in any direction. This is made possible with the help of a language called *Hypertext Markup Language* (HTML). Web pages contain HTML tags that enable us to find, retrieve, manipulate and display documents worldwide.

Java was meant to be used in distributed environments such as Internet. Since, both the Web and Java share the same philosophy, Java could be easily incorporated into the Web system. Before Java, the World Wide Web was limited to the display of still images and texts. However, the incorporation of Java into Web pages has made it capable of supporting animation, graphics, games, and a wide range of special effects.

9. What are Web Browsers? (Oct 2012)

The Internet is a vast sea of information represented in many formats and stored on many computers. A large portion of the Internet is organized as the World Wide Web which uses hypertext. Web browsers are used to navigate through the information found on the net. They allow us to retrieve the information spread across the Internet and display it using the HyperText Markup Language (HTML).
Examples of Web Browsers are: HotJava, Netscape Navigator, Internet Explorer.

10. What is the task of main method in Java? (Oct 2010)

Let us consider the sample Java code:

```
class SampleOne
{
  public static void main (String args[ ])
  {
    System.out.println ("Java is better than C++.");
  }
}
```

Class Declaration

The first line

```
class SampleOne
```

declares a class, which is an object-oriented construct. As stated earlier, Java is a true object-oriented language and therefore, ***everything*** must be placed inside a class. **class** is a keyword and declares that a new class definition follows. **SampleOne** is a Java *identifier* that specifies the name of the class to be defined.

Opening Brace

Every class definition in Java begins with an opening brace "{" and ends with a matching closing brace "}", appearing in the last line in the example. This is similar to C++ class construct. (*Note that a class definition in C++ ends with a semicolon.*)

The Main Line

The third line

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

defines a method named **main**. Conceptually, this is similar to the **main( )** function in C/C++. Every Java application program must include the **main( )** method. This is the starting point for the interpreter to begin the execution of the program. A Java application can have any number of classes but ***only one*** of them must include a **main** method to initiate the execution.

This line contains a number of keywords, **public, static** and **void**.

Public:	The keyword **public** is an access specifier that declares the **main** method as unprotected and therefore making it accessible to all other classes. This is similar to the C++ **public** modifier.
Static:	Next appears the keyword **static**, which declares this method as one that belongs to the entire class and not a part of any objects of the class. The **main** must always be declared as **static** since the interpreter uses this method before any objects are created.
Void:	The tyep modifier **void** states that the **main** method does not return any value (but simply prints some text to the scree.)

All parameters to a method are declared inside a pair of parentheses. Here, **String args[ ]** declares a parameter named **args**, which contains an array of objects of the class type **String**.

The Output Line

The only executable statement in the program is

```
System.out.println ("Java is better than C++.");
```

This is similar to the **printf( )** statement of C or **cout <<** construct of C++. Since Java is a true object oriented language, every method must be part of an object. The **println** method is a member of the **out** object, which is a static data member of **System** class.

This line print the string
Java is better than C++.
to the screen. The method **println** always appends a newline character to the end of string. This means that any subsequent output will start on a new line. Note the semicolon at the end of the statement. *Every Java statement must end with a semicolon.*

11. Explain the structure of Java Program. (Oct 2010)

Why do we need import statement? (Oct 2011)

Java Program Structure: A Java program may contain many classes of which only one class defines a main method. Classes contain data members and methods that operate on the data members of the class. Methods may contain data type declarations and executable statements. To write a Java program, we first define classes and then put them together. A Java program may contain one or more sections as shown below:

Documentation Section	Suggested
Package Statement	Optional
Import Statements	Optional
Interface Statements	Optional
Class Definitions	Optional

main Method Class
{
Main Method Definition
}
	Essential

Documentation Section: The documentation section comprises a set of comment lines giving the name of the program, the author and other details. Comments explain *why* and *what* of classes and *how* of algorithms. This would greatly help in maintaining the program. Java uses a new style of comment /**....*/ known as *documentation comment.* This form of comment is used for generating documentation automatically.

Package Statement: The first statement allowed in a Java file is a *package* statement. This statement declares a *package* name and informs the compiler that the classes defined here belong to this package. Example:

package student;

The package statement is optional. That is, our classes do not have to be part of package.

Import Statements: The next thing after a package statement (but before any class definitions) may be a number **of** *import* statements. This is similar to the *include* statement in C. Example:

import student.test;

This statement instructs the interpreter to load the *test* class contained in the package *student***.** Using import statements, we can have access to classes that are part of other named packages.

Interface Statements: An interface is like a class but includes a group of method declarations. This is also an optional section and is used only when we wish to implement the multiple inheritance features in the program.

Class Definitions: A Java program may contain multiple class definitions. Classes are the primary and essential elements of a Java program. These classes are used to map the objects of real-world problems. The number of classes used depends on the complexity of the problem.

Main Method Class: Since every Java stand-alone program requires a **main** method as its starting point, this class is the essential part of a Java program. A simple Java program may contain only this part. The **main** method creates objects of various classes and establishes communications between them. On reaching the end **of main,** the program gets terminated.

12. What is a Token? List the various tokens supported by Java. (Mar 2011) (Mar 2012)

Java Tokens: Tokens are the ones which are used to prepare a program. The statements of a java program contain expressions, which describe the actions carried out on data. Smallest individual units in a program are known as *tokens.* The compiler recognizes them for building up expressions and statements.

In simple terms, a Java program is a collection of tokens, comments and white spaces. Java language includes five types of tokens. They are:
1. Reserved Keywords
2. Identifiers
3. Literals
4. Operators
5. Separators

Java Character Set: The smallest units of Java language are the characters used to write Java tokens. These characters are defined by the *Unicode* character set. The Unicode is a 16-bit character coding system and currently supports more than 34,000 defined characters derived from 24 languages.

Keywords: Keywords are an essential part of a language definition. They implement specific features of the language. Java language has reserved 50 words as keywords. Since keywords have specific meaning in Java, we cannot use them as names for variables, classes, methods and so on. All keywords are to be written in lower-case letters.

Identifiers: Identifiers are user defined tokens. They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program. Java identifiers follow the following rules:

1. They can have alphabets, digits, and the underscore and dollar sign characters.
2. They must not begin with a digit.
3. Uppercase and lowercase letters are distinct.
4. They can be of any length.

Literals: Literals in Java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables. Java language specifies five major types of literals. They are:

1. Integer literals
2. Floating point literals
3. Character literals
4. String literals
5. Boolean literals

Operators: An operator is a symbol that takes one or more arguments and *operates* on them to produce a result. Operators are of many types such as arithmetic, relational and etc.

Separators: Separators are symbols used to indicate where groups of code are divided and arranged. They basically define the shape and function of our code. They include symbols like braces, comma and etc.

13. Explain Java Statements.

A statement is an executable combination of tokens ending with a semicolon (;) mark. Statements are usually executed in sequential order. However, it is possible to control the flow of execution, if necessary, using special statements. Java implements several types of statements as illustrated in the following Table.

Empty Statement	These do nothing and are used during program development as a place holder.
Labeled Statement	Any Statement may begin with a label. Such labels must not be keywords, already declared local variables or previously used labels in this module. Labels in Java are used as the arguments of Jump statements.

Expression Statement	Most statements are expression statements. Java has seven types of Expression statements Assignment, Pre-Increment, Pre-Decrement, Post-Increment, Post-Decrement, Method Call and Allocation Expression.
Selection Statement	These select one of several control flows. There are three types of selection statements in Java if, if-else, and switch.
Iteration Statement	These specify how and when looping will take place. There are three types of iteration statements; while, do and for.
Jump Statement	Jump Statements pass control to the beginning or end of the current block, or to a labeled statement. Such labels must be in the same block, and continue labels must be on an iteration statement. The four types of Jump statement are *break, continue, return and throw.*
Synchronization Statement	These are used for handling issues with multithreading.
Guarding Statement	Guarding statements are used for safe handling of code that may cause exceptions (such as division by zero). These statements use the keywords try, catch, and finally.

14. How to implement a Java Program? (Mar 2010)

Implementation of a Java application program involves a series of steps. They include:

1. Creating the program
2. Compiling the program
3. Running the program

Creating the Program: We can create a program using any text editor like notepad. But, we need to make sure that the program name should be saved with the extension ".java". Advisably, the name of the program should match with the name of the public class or a class that has main() method.

Example: assumed that the program name is "Test.java"

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

Compiling the program: Java program involves two steps for its execution: one compilation and the other execution. We use "javac" tool to compile java program. This yields byte code file whose extension is ".class". This is an intermediate file and not directly executable.

Syntax: *javac Test.java*

Running the Program: Now, java program is interpreted by the tool "java" which as an interpreter. This is different for different machines. This tool generates executable code for the machine in use. Thus one can view the output of the java program.

Syntax: *java Test*

The complete steps to run a java program are listed below:

1. Type the program in the DOS editor or notepad. Save the file with a .java extension.
2. The file name should be the same (advisably) as the class, which has the main method.
3. To compile the program, using javac compiler, type the following on the command line: *Example:* javac Test.java
4. After compilation, run the program using the Java interpreter.
Example: java Test
5. The program output will be displayed on the command line.

15. Write short notes on JVM. (Mar 2011) (Mar 2012) (Oct 2012)

All language compilers translate source code into *machine code* for a specific computer. But, Java compiler produces an intermedia code known as *bytecode* for a machine that does not exist. This machine is called the *Java Virtual Machine* and it exists only inside the computer memory. It is a simulated computer within the computer and does all major functions of a real computer

The virtual machine code is not machine specific. The machine specific code is generated by the Java Interpreter by acting as an intermediary between the virtual machine and the real machine. The Java API acts as the intermediary between the operating system and the Java Framework.

The virtual machine code is not machine specific. The machine specific code (known as machine code) is generated by the Java interpreter by acting as an intermediary between the virtual machine and the real machine. The interpreter is different for different machines.

16. What are Command-Line Arguments? How are they used in Java? (Oct 2011) (Mar 2012)

Command line arguments are the parameters passed to the executable java program in the form of strings. Command line arguments are collected in main function itself. The argument is an array of strings and can collect the list of strings. File name is not included in the list of arguments as in C language. This is very useful to receive quick inputs.

```
class Test
{
public static void main(String args[])
{
System.out.println("Number of arguments="+as.length);
for(int i=0;i<as.length;i++)</as.length;i++)
System.out.println(as[i]);
}
}
```

Execute the above program as follows.

java Test 1 2 3

The above command produces the following output:

Number of arguments=3

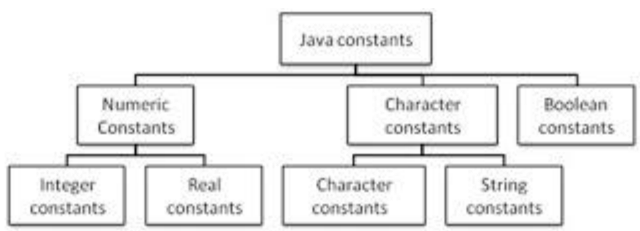
1

2

3

17. Difference between Constant and Variable. (Mar 2010).

CONSTANTS: Constants are values which can never be changed. Constants can be directly used in expressions or they are assigned to variables.



Integer constants: An integer type of variables can hold integer constants. There are three types of integers, namely, decimal, octal and hexadecimal integers.

Ex: int x=30

Real (Float) constants: A real variable can hold float constants. They are whole numbers followed by a decimal point and fractional part. They may be expressed in scientific (exponential) notation.

Ex: float pi=3.14, x=0.64e4

Char constants: A character variable can hold only one character. Character constant must be enclosed in single quotes.

Ex: char ch= 'Q'

String constants: String constant is a character array that can hold string values. Strings must be enclosed in double quotes.

Ex: String s= “Hello”, s1=”1234”;

Boolean (logical) constants: Boolean constants are *true* and *false*. Unlike java, languages like C, C++ use integers to represent true or false.

Ex: boolean x=true

Backslash Character Constants: These are used in output for formatting. All of these are similar to the ones which are available in C language. The constants and their meanings are listed below.

CONSTANT	MEANING
\b	backspace
\f	form feed
\n	new line
\r	carriage return
\t	horizontal tab
\’	single quote
\”	double quote
\\	backslash

VARIABLES: A *variable* is an identifier that denotes a storage location used to store a data value. Unlike constants, a variable may take different values in different times during the execution of the program. A variable name can be chosen by the programmer in a meaningful way so as to reflect what it represents in the program. Some examples of variable names are:

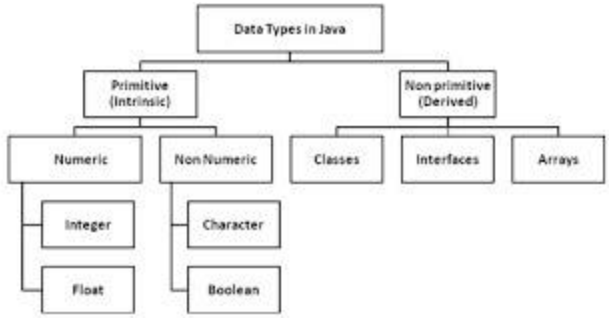
1. average
2. height
3. total_height
4. classAverage

A variable names may consist of alphabets, digits, the underscore(_) and dollar characters, subject to the following conditions:

1. They must not begin with a digit.
2. Uppercase and lowercase are distinct. This means that the variable **Total** is not the same as total or **TOTAL.**
3. It should not be a keyword.
4. White space is not allowed.
5. Variable names can be of any length.

18. What are the different data types in Java? (Oct 2010) (Oct 2011) (Oct 2012)

Data-type: Data-type specifies what type of value a variable can store. It also specifies its size. Every variable in Java has a data-type. Java language is rich in its data types. The categories of different data-types are shown below. The two main types are Primitive and non-primitive types.



1. Integer type: An integer type of variables can hold integer constants. Java supports four types of integer data types. They are *byte, short, int* and *long.* The following table shows size in bytes and range of values each data type can represent. Integer can be converted to long type by typing 'L' after them.

TYPE	MEMORY (bytes)	VALUES
byte	1	-128 to 127
short	2	-32768 to 32767
int	4	-2^{31} to $2^{31}-1$
long	8	-2^{63} to $2^{63}-1$

2. Floating point type: This data type is used to represent numbers containing fractional values. They are two types of data-types under this category, namely, float and double. float type number are single precision numbers while double type number are double precision numbers. By default float are double precision and can converted to single precision by adding 'f' at the end. Double precision types are used when greater precision is required in storage of floating point. Floating point data types have special NaN (Not a Number) value for undefined numbers like 0/0 etc.

TYPE	SIZE	MAXIMUM VALUE	MINIMUM VALUE
float	4 bytes	3.4e-038	3.4e+038
double	8 bytes	1.7e-308	1.7e+308

3. Character type: Java provides a data type called char to hold character values. It is of size 2 bytes and assumes single character. The character value must be enclosed in single quotes.

4. Boolean type: It can have only two values, either 'true' or 'false'. Its keyword is 'boolean' and use only 1 bit. All comparison operators return boolean type values. The words 'true' and 'false' cannot be used as identifiers.

19. What is the scope of a variable? (Mar 2010)

Scope of a variable determines the area in which a variable can be accessed.

Java variables are actually classified into three kinds:

1. instance variables
2. class variables
3. local variables

Instance and class variables are declared inside a class. They are accessible throughout the class. If these are 'public' type, they are also accessible outside but with the objects or with class-name when it is a class variable. Whereas local variables are accessible only in the methods in which they are declared.

Instance Variables: Instance variables are created when the objects are instantiated and therefore they are associated with the objects. They take different *values* for each object.

Class variables are global to a class and belong to the entire set of objects that class creates. Only one memory location is created for each class variable. These are basically static data members of a class. They take common value for all the objects.

Local variables: Variables declared and used inside methods are called *local variables.* They are called so because they are not available for use outside the method definition. Local variables can also be declared inside program blocks that are defined between an opening brace '{' and a closing brace '}'. These variables are visible to the program only from the beginning of its program block to the end of the program block. When the program control leaves a block, all the variables in the block will cease to exist.

20. Explain Symbolic Constants. (Mar 2011) (Mar 2012)

It is necessary to use certain unique constants in a program. These constants may appear repeatedly in a number of places in the program. One example of such constant is 3.142, representing the value of the mathematical constant "pi". Another example is the total number of students that may be used for calculating the class total, class average, standard deviation, etc. We face two problems in the subsequent use of such programs. They are:

1. Problem in modification of the program.
2. Problem in understanding the program.

Modifiability: We may like to change the value of "pi" from 3.142 to 3.14159 to improve the accuracy of calculations. For this, we will have to search throughout the program and explicitly change the value of the constant wherever it has been used. If any value is left unchanged, the program may produce disastrous outputs.

Understandability: When a numeric value appears in a program, its use is not always clear, especially when the same value; means different things in different places. Assignment of a symbolic name to such constants frees us from these problems.

A constant is declared as follows:

1. Symbolic names take the same form as variable names. But, they are written in CAPITALS to visually distinguish them from normal variable names. This is only a convention, not a rule.

2. After declaration of symbolic constants, they should not be assigned any other value within the program by using an assignment statement. For example, STRENGTH = 200; is illegal.

3. Symbolic constants are declared for types. This is not done in C and C++ where symbolic constants are defined using the # define statement.

4. They can NOT be declared inside a method. They should be used only as class data members in the beginning of the class.

Example:
final float PI=3.14159
final int STRENGTH=50

21. Explain Type Casting. (Mar 2012)

Type casting is a process to convert one data type to another. It makes the variable compatible temporarily. We often encounter situations where there is a need to store a value of one type into a variable of another type. In such situations, we must cast the value to be stored by preceding it with the type name in parentheses. Casting is often necessary when a method returns a type different than the one we require.

Four integer types can be cast to any other type except boolean. Casting into a smaller type may result in a loss of data. Similarly, the **float and double** can be cast to any other type except boolean. Again, casting to smaller type can result in a loss of data. Casting a floating point value to an integer will result in a loss of the fractional part. If the following order is used for type casting, it guarantees in no loss of information.

byte → char → short → int → long → float → double

for example

- *byte* value can be converted to *int* or *double*
- *int* value can be converted to *long* or *float*

The syntax for casting is: *type variable1 = (type) variable2;*

Type casting can be of two types:

Implicit type Casting (Automatic Conversion): when constants and variables of different types are mixed in an expression they are converted to the same type. This conversion is done implicitly by the C compiler. The C compiler converts all the operands to the type of the largest operand. For example if an expression involves int and float, int type gets converted to float type.

Example:

float x=32;

The value of the x will be '32.0', because of implicit conversion of value from int to float

Explicit type Casting: if a user forcefully changes the data type into other allowable data type it is said to be explicit type casting.

Example:

int to float:
float x;
x=5/2 /* value of x will be 2.0 */
x=(float)5/2; /* value of x will be 2.5 */

float to int:

int x;
x= 3.2 /* causes error */
x=(int)3.2 /* x will be 3 */