JA111

Day-9

Package

- Packages help in organizing our Java files.
- It is a mechanism of grouping similar types of classes and interfaces collectively based on functionality.

To create a package

1. File->New->Package; type package name
2. To make a file part of the package, the first line in the file will be-
   package package-name;

Advantages

1. Modularization
2. Namespace Management
3. Controlled Access
4. Data Encapsulation

Types of packages

1. Built-in Packages
2. User-Defined Packages

- If a package statement is not used, then the Java files will be placed in the current default package.
- The naming convention for packages is to write the complete package in lowercase.

Using class of a package in another

- The import statement is used to access classes and interfaces of a package in another package. It's general form is as follow-

```
import package-name.class-name;  or import package-name.*;
```

- Another way to access classes and interfaces of a package in another package is by using the fully qualified name.

Access Specifier

	private	default	protected	public
Same class	Yes	Yes	Yes	Yes
Same package Sub class	No	Yes	Yes	Yes
Same package other class	No	Yes	Yes	Yes
Different package sub class	No	No	Yes	Yes
Different package other class	No	No	No	Yes

- The static import allow to import only static member(s) of a class so no need to use class name to access static members of class. Its overuse is not recommended

Abstract class

- An abstract class is one that has abstract methods (not necessarily). To create an abstract class, we have to use abstract keyword before the class-name.

```
abstract class class-name{
  ....
}
```

Think!
Difference between abstract class and concrete class

Tip: An abstract class can have variables, concrete methods and constructors also.

- An abstract method is one that has only prototype but its body is missing. To create an abstract method, we have to use abstract keyword

```
abstract return-type method-name(para-list);
```

- It is not possible to create an object of abstract class but it is perfectly okay to create reference variables.
- For a subclass of abstract class, it is mandatory to provide definition of all abstract methods, if subclass fails to do the same then it must also be an abstract class.
- A reference variable of abstract class can point to the object of its implementing class and when abstract method is called using that reference variable then calling will take place on the behalf of dynamic method dispatch so abstract class uses run time polymorphism

Variable arguments (Var-Args) in Java

- We can define them using a standard type declaration, followed by an ellipsis:

```
access-specifier return-type method-name(data-type… var-name) {
    // …
}
```

- Following points to be remember in case of the variable arguments-
  - Using variable arguments a method can handle an arbitrary number of parameters (using an array internally).
  - Each method can only have one variable argument parameter
  - The variable argument argument must be the last parameter
  - The advantage of is that variable arguments help us avoid writing such boilerplate code

Enumerations (Enums)

- Enumerations allows us to limit the selection within a set of values; this can help us to keep away from random input of user
- Enumerations are group of named constants. All Enumerations implicitly extend the java.lang.Enum class. A programmer can define constructors, methods and variables, inside Enumerations.
- The Enumerations fields are implicitly static and final, and hence are constant during compile time. But they are instances of their enum type, constructed when the enum type is referenced for the first time. The Enumerations fields are written in capital letter according to java convention.
- Enum can be written inside and outside class but not inside a method.
- Enum variables can be used in an if statement or switch statement.
- A static method called values() is automatically generated by the Java compiler for each enum. The values() method returns an array of all the constant values defined inside the enum. ordinal() method can be used to display values assigned to enum constants.
- To create an enum

```
public enum enum_name { constant1, constant2, ..., constant n }
```

- To create variable of class enum

```
access-modifier enum-name ref-name [ = enum-name.constant-name];
```