Creating User-defined Generic Classes and Methods

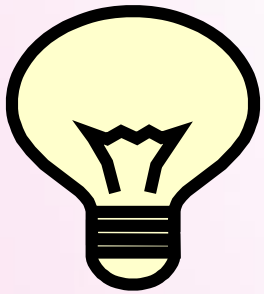
- Scenario:

A developer

Needs to develop a class with a method, which can return various objects, such as `Integer`, `String`, and `Double`.

Creating User-defined Generic Classes and Methods (Contd.)

- Scenario (Contd.):



To implement the functionality, a developer needs to create generic classes and methods.

Creating Generic Classes

- Generics enable you to generalize classes.
- In the declaration of a generic class, the name of the class is followed by a type parameter section.
- The type parameter section is represented by angular brackets, < >, that can have one or more types of parameters separated by commas.
- The following syntax is used to create generic classes:

```
class [ClassName]<T>
{

}
```

- You can create a generic class by using the following code snippet:

```
class GenericClassDemo<T>
{

}
```

Creating Generic Classes (Contd.)

- In case you want to create a generic class to set and get the `Integer` and `String` values, you can use the following code:

```
public class GenericClassDemo<T>
{
    private T t;

    public void setValue(T t)
    {
        this.t = t;
    }
    public T getValue()
    {
        return t;
    }
```

Creating Generic Classes (Contd.)

```
        public static void main(String[] args)
        {
            GenericClassDemo<Integer> iobj = new
    GenericClassDemo<Integer>();
            iobj.setValue(10);
            System.out.println(iobj.getValue());

            GenericClassDemo<String> sobj = new
    GenericClassDemo<String>();
            sobj.setValue("Ten");
            System.out.println(sobj.getValue());
        }
    }
```

Creating a Generic Method

- In the generic class, a method can use the type parameter of the class, which automatically makes the method generic.
- Consider the following code of the generic method inside the generic class:

```
public class GenericClassDemo<T>
{
    private T t;
    public void setValue(T t)
            {
                    this.t = t;
            }
    public T getValue()
            {
                    return t;
            }
 }
```

Creating a Generic Method (Contd.)

- The declaration of a generic method contains the type parameter that is represented by angular brackets, < >.
- The following syntax is used to create a generic method:

```
public <Type Parameter> [Return Type]
[MethodName](Argument list…)
{
   }
```

- You can create a generic method, as shown in the following code snippet:

```
public <T> T showValue(T val)
{
}
```

Creating a Generic Method (Contd.)

- Consider the following code of the generic method inside the non-generic class:

```
public class GenericMethodDemo
{
   public <M> M display(M val)
   {
       return val;
   }

   public static void main(String[] args)
   {
       GenericMethodDemo obj = new
GenericMethodDemo();
```

Creating a Generic Method (Contd.)

```
        System.out.println("The generic method is
called with String value: " + obj.display("Test"));
        System.out.println("The generic method is
called with Double value: " + obj.display(7.5));
        System.out.println("The generic method is
called with Boolean value: " + obj.display(true));
        System.out.println("The generic method is
called with Integer value: " + obj.display(10));

    }
}
```

Summary

- In this session, you learned that:
  - Quantifiers are of three types:
    - Greedy
    - Reluctant
    - Possessive
  - Localization is a process of customizing the application to a specific locale and culture.
  - Localization can be implemented on different types of data, such as date, currency, and text.
  - To localize different types of data, it is necessary to determine the language and country.
  - To determine the language, Java provides a predefined set of language codes, such as `zh` for Chinese and `en` for English.
  - To work with localization, the `Locale` class of the `java.util` package is used.

Summary (Contd.)

- To localize the date, we need to use the various date formats. To determine the date format according to the locale, you can use the `java.text.DateFormat` class.
- To localize the currency, we need to use various currency formats. To determine the currency format according to the locale, you can use the `java.text.NumberFormat` class.
- To localize text, we need to use the resource bundles.
- A resource bundle is a property file that contains the locale specific data. To work with data stored in a resource bundle, we need to use the `ResourceBundle` class.
- To have a locale-specific object, a program needs to load the `ResourceBundle` object by using the `getBundle()` method.
- Generics enable you to generalize classes, which mean that a reference variable, an argument of a method, and a return type can be of any type.

Summary (Contd.)

- The type parameter section is represented by angular brackets, < >, that can have one or more types of parameters separated by commas.
- In the generic class, a method can use the type parameter of the class, which automatically makes the method generic.
- You can declare a generic method, which contains its own one or more type parameters.
- The declaration of a generic method contains the type parameter that is represented by angular brackets, < >.