Core Java

Day 10 Agenda

- OOP - Interview Question Discussion
- Generic Programming
  - Using java.lang.Object
  - Generic Classes
  - Advantages of Generics
  - Bounded & Unbounded generic types
  - Upper & Lower bounded generic types

Generic Programming

- Code is said to be generic if same code can be used for various (practically all) types.
- Best example:
  - Data structure e.g. Stack, Queue, Linked List, ...
  - Algorithms e.g. Sorting, Searching, ...
- Two ways to do Generic Programming in Java
  - using java.lang.Object class -- Non typesafe
  - using Generics -- Typesafe

Generic Programming Using java.lang.Object

```
```Java
class Box {
    private Object obj;
    public void set(Object obj) {
        this.obj = obj;
    }
    public Object get() {
```

SUNBEAM INFOTECH
```

```
        return this.obj;
    }
}
```
```Java
Box b1 = new Box();
b1.set("Nilesh");
String obj1 = (String)b1.get();
System.out.println("obj1 : " + obj1);

Box b2 = new Box();
b2.set(new Date());
Date obj2 = (Date)b2.get();
System.out.println("obj2 : " + obj2);

Box b3 = new Box();
b3.set(new Integer(11));
String obj3 = (String)b3.get();  // ClassCastException
System.out.println("obj3 : " + obj3);
```

Generic Programming Using Generics

- Added in Java 5.0.
- Similar to templates in C++.
- We can implement
    - Generic classes
    - Generic methods
    - Generic interfaces

Advantages of Generics

- Stronger type checking at compile time i.e. type-safe coding.
- Explicit type casting is not required.

SUNBEAM INFOTECH

- Generic data structure and algorithm implementation.

Generic Classes

- Implementing a generic class

```
class Box<TYPE> {
    private TYPE obj;
    public void set(TYPE obj) {
        this.obj = obj;
    }
    public TYPE get() {
        return this.obj;
    }
}
```

```
Box<String> b1 = new Box<String>();
b1.set("Nilesh");
String obj1 = b1.get();
System.out.println("obj1 : " + obj1);

Box<Date> b2 = new Box<Date>();
b2.set(new Date());
Date obj2 = b2.get();
System.out.println("obj2 : " + obj2);

Box<Integer> b3 = new Box<Integer>();
b3.set(new Integer(11));
String obj3 = b3.get();  // Compiler Error
System.out.println("obj3 : " + obj3);
```

- Instantiating generic class

SUNBEAM INFOTECH

```
Box<String> b1 = new Box<String>(); // okay

Box<String> b2 = new Box<>(); // okay -- type inference

Box<> b3 = new Box<>(); // compiler error -- type must be given while creating generic class reference
                        // cannot be auto-detected

Box<Object> b4 = new Box<String>(); // compiler error

Box b5 = new Box(); // okay -- internally considered Object type -- compiler warning "raw types"

Box<Object> b6 = new Box<Object>(); // okay -- Not usually required/used
```

Generic types naming convention

1. T : Type
2. N : Number
3. E : Element
4. K : Key
5. V : Value
6. S,U,R : Additional type param

Bounded generic types

- Bounded generic param restricts data type that can be used as type argument.
- Decided by the developer of the generic class.

```
class Box<T extends Number> {
    private T obj;
    public T get() {
        return this.obj;
```

SUNBEAM INFOTECH

```
    }
    public void set(T obj) {
        this.obj = obj;
    }
}
```

- The Box<> can now be used only for the classes inherited from the Number class.

```
Box<Number> b1 = new Box<>(); // okay
Box<Boolean> b2 = new Box<>(); // error
Box<Character> b3 = new Box<>(); // error
Box<String> b4 = new Box<>(); // error
Box<Integer> b5 = new Box<>(); // okay
Box<Double> b6 = new Box<>(); // okay
Box<Date> b7 = new Box<>(); // error
Box<Object> b8 = new Box<>(); // error
```

Unbounded generic types

- Unbounded generic type is indicated with wild-card "?".
- Can be given while declaring generic class reference.

```
class Box<T> {
    private T obj;
    public Box(T obj) {
        this.obj = obj;
    }
    public T get() {
        return this.obj;
    }
    public void set(T obj) {
```

SUNBEAM INFOTECH

```
        this.obj = obj;
    }
}
```

```
public static void printBox(Box<?> b) {
    Object obj = b.get();
    System.out.println("Box contains: " + obj);
}
```

```
Box<String> sb = new Box<String>("DAC");
printBox(sb); // ??
Box<Integer> ib = new Box<Integer>(100);
printBox(ib); // ??
Box<Date> db = new Box<Date>(new Date());
printBox(db); // ??
Box<Float> fb = new Box<Float>(200.5f);
printBox(fb); // ??
```

Upper bounded generic types

- Generic param type can be the given class or its sub-class.

```
public static void printBox(Box<? extends Number> b) {
    Object obj = b.get();
    System.out.println("Box contains: " + obj);
}
```

SUNBEAM INFOTECH

```
Box<String> sb = new Box<String>("DAC");
printBox(sb); // ??
Box<Integer> ib = new Box<Integer>(100);
printBox(ib); // ??
Box<Date> db = new Box<Date>(new Date());
printBox(db); // ??
Box<Float> fb = new Box<Float>(200.5);
printBox(fb); // ??
```

Lower bounded generic types

- Generic param type can be the given class or its super-class.

```
public static void printBox(Box<? super Integer> b) {
    Object obj = b.get();
    System.out.println("Box contains: " + obj);
}
```

```
Box<String> sb = new Box<String>("DAC");
printBox(sb); // ??
Box<Integer> ib = new Box<Integer>(100);
printBox(ib); // ??
Box<Date> db = new Box<Date>(new Date());
printBox(db); // ??
Box<Float> fb = new Box<Float>(200.5f);
printBox(fb); // ??
Box<Number> nb = new Box<Number>(null);
printBox(nb); // ??
```

SUNBEAM INFOTECH

Assignment

1. Copy Person class and inherited classes (Employee, Salesman, Manager, SalesManager) from previous assignment/classwork. Implement generic class Box so that it can store any Person in it. How to get total salary of the Employee in Box?

2. Copy Displayable, Person, and Date classes from previous assignment/classwork. Complete the following non-generic methods with appropriate parameters and call them from main().

```
public static void printDisplayableBox(______ b) {
    b.get().show();
}
```

```
public static void printAnyBox(______ b) {
    System.out.println(b.get().toString());
}
```

3. **OPTIONAL - OOP ASSIGNMENT** Create an abstract Player class with id, name, age, and matchesPlayed as fields. Create a Batter interface with methods like getRuns(), getAverage(), and getStrikeRate(). Creater a Bowler interface with methods like getWickets(), and getEconomy(). Create a class Cricketer inherited from Player as well as Batter and Bowler interfaces. In all classes write appropriate constructors, getter/setters, accept(), toString(), and equals() methods. In main(), create a team (array) of 11 players and input their details from end user. Create a new (utilility) class Players that contains static methods to count number of batters, number of bowlers, total batter runs, total bowler wickets, return a batter with maximum runs, and return a bowler with maximum wickets. Following code snippets can be helpful.

```
class Players {
    public static int batterTotalRuns(Player[] arr) {
        int runs = 0;
        for(Player p:arr) {
            if(p instanceof Batter) {
                Batter b = (Batter)p;
                runs = runs + b.getRuns();
```

SUNBEAM INFOTECH

```
            }
        }
        return runs;
    }
    public static int bowlerTotalWickets(Player[] arr) {
        // ...
    }
    public static int countBatters(Player[] arr) {
        // ...
    }
    public static int countBowlers(Player[] arr) {
        // ...
    }
    public static Player maxRunBatter(Player[] arr) {
        // ...
    }
    public static Player maxWicketBowler(Player[] arr) {
        // ...
    }
}
```

```
class Program {
    public static void main(String[] args) {
        // ...
        Player[] team = new Player[11];
        for(int i=0; i<team.length; i++) {
            System.out.print("Enter 1 for batter and 2 for bowler: ");
            int choice = sc.nextInt();
            if(choice == 1)
                arr[i] = new Batter();
            else
                arr[i] = new Bowler();
            arr[i].accept();
        }
```

SUNBEAM INFOTECH

```
        for(Player p:team)
            System.out.println(p.toString());

        int totalRuns = Players.batterTotalRuns(team);
        System.out.println("Total runs of all batters: " + totalRuns);

        // ...
    }
}
```

SUNBEAM INFOTECH