1. Program to Display Current Date and Time in Java

You can use `LocalDateTime` and `DateTimeFormatter` from the `java.time` package to display the current date and time:

Eg:-

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class CurrentDateTime {
    public static void main(String[] args) {
        // Get current date and time
        LocalDateTime current = LocalDateTime.now();

        // Format date and time
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
        String formatted = current.format(formatter);

        // Display current date and time
        System.out.println("Current Date and Time: " + formatted);
    }
}
```

Eg:-

2. Program to Convert a Date to a String in the Format "MM/dd/yyyy"

You can use `SimpleDateFormat` from the `java.text` package to convert a date to a string:

Eg:-

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateToString {
    public static void main(String[] args) {
        // Create a date object
        Date date = new Date();

        // Define date format
        SimpleDateFormat formatter = new SimpleDateFormat("MM/dd/yyyy");

        // Convert date to string
        String strDate = formatter.format(date);

        // Display the formatted date string
```

```
        System.out.println("Formatted Date: " + strDate);
    }
}
```
Eg:-

3. Difference Between Collections and Streams in Java

Eg:-CollectionsEg:- in Java provide a way to store and manipulate a group of objects.
Eg:-StreamsEg:-, on the other hand, represent a sequence of elements supporting sequential and parallel aggregate operations.

Eg:-Key Differences:Eg:-
- Eg:-Collections:Eg:- Store and manipulate data.
- Eg:-Streams:Eg:- Perform operations on data without storing it.

Eg:-Example:Eg:-

Eg:-
```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class CollectionsVsStreams {
    public static void main(String[] args) {
        // Using Collections (List)
        List<String> names = Arrays.asList("Alice", "Bob", "Charlie", "David");
        names.forEach(System.out::println); // Iterating through the collection

        // Using Streams
        List<String> filteredNames = names.stream()
                                          .filter(name -> name.startsWith("A"))
                                          .collect(Collectors.toList());
        filteredNames.forEach(System.out::println); // Output: Alice
    }
}
```
Eg:-

4. Enums in Java

Eg:-EnumsEg:- are a special class that represents a group of constants (unchangeable variables). They are often used when we have a fixed set of related values.

Eg:-Example:Eg:-

Eg:-

```
enum Day {
   SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
}

public class EnumExample {
   public static void main(String[] args) {
      Day day = Day.MONDAY;
      System.out.println("The day is: " + day);

      // Looping through the enum values
      for (Day d : Day.values()) {
         System.out.println(d);
      }
   }
}
```

Eg:-

5. Built-In Annotations in Java

Java has several built-in annotations that provide information to the compiler or are used to enforce certain behaviors in code.

Eg:-Common Built-In Annotations:Eg:-

1. `@Override` - Indicates that a method is overriding a method from a superclass.
2. `@Deprecated` - Marks a method or class as deprecated and not recommended for use.
3. `@SuppressWarnings` - Instructs the compiler to suppress specific warnings.
4. `@FunctionalInterface` - Marks an interface as a functional interface (an interface with a single abstract method).
5. `@SafeVarargs` - Suppresses unsafe operations warning when using varargs.

Eg:-Example:Eg:-

Eg:-

```
class Parent {
   @Deprecated
   void display() {
      System.out.println("This method is deprecated.");
   }
}

public class AnnotationExample extends Parent {
   @Override
```

```
    void display() {
        System.out.println("Overridden method.");
    }

    @SuppressWarnings("unchecked")
    public static void main(String[] args) {
        AnnotationExample obj = new AnnotationExample();
        obj.display();  // This will call the overridden method
    }
}
```

Eg:-

These concepts cover a wide range of Java fundamentals from date manipulation to enums and annotations! Let me know if you need more examples or details on any topic.