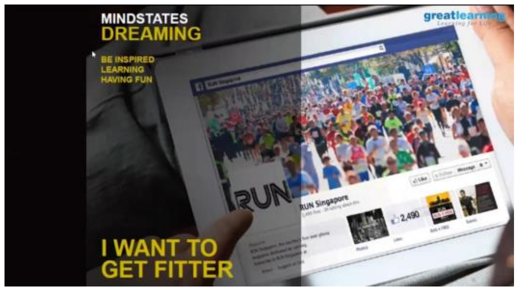
DAILY ASSESSMENT FORMAT

Date:	**15/06/2020**	**Name:**	**Lavanya B**
Course:	**Great learning**	**USN:**	**4al17ec043**
Topic:	**Digital marketing**	**Semester & Section:**	**6th A**
Github Repository:	**Lavanya-B**		

FORENOON SESSION DETAILS

Image of session

greatlearning
Learning for Life
← Go Back to Introduction to Digital Marketing
Course Content
Consumer Journey of Today



Report

Digital marketing

Digital marketing is the component of marketing that utilizes internet and online based digital technologies such as desktop computers, mobile phones and other digital media and platforms to promote products and services.

New medias

New media as computer technology used as a distribution platform – New media are the cultural objects which use digital computer technology for distribution and exhibition. Eg: Internet, Web sites, computer multimedia, Blu-ray disks etc.

Understanding brand purpose

Brand purpose is the reason for the brand to exist beyond making money. If you want a really powerful brand purpose, it needs to relate to the product or service itself. For example, if you're in the educational sector, your purpose might help children and shape their future.

1. Start with ideology
2. Context is crucial
3. Offer real value
4. Love the haters
5. Be consistent

Facebook marketing

Facebook marketing refers to creating and actively using a Facebook page as a communications channel to maintain contact with and attract customers. Facebook actively provides for this, allowing users to create individual profiles or business pages for companies, organizations, or any group attempting to develop a fan.

Facebook marketing used by

- Brands
- Local business
- Personalities
- Non-profit organization

Purpose of Facebook marketing

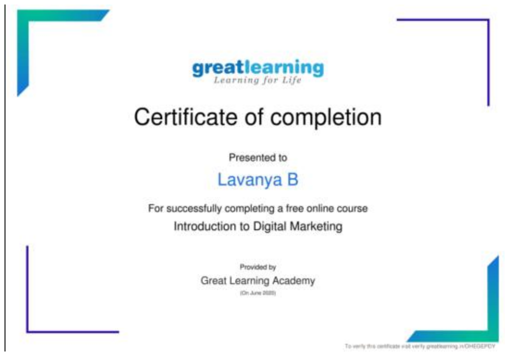
A major benefit of Facebook advertising is its ability to reach your exact audience. Facebook is the most targeted form of advertising. You can advertise to people by age, interests, behavior, and location. If you really know your customers, you can use Facebook advertising to engage them.

Types of Ads

The advertisement include video, offers, leads, canvas, carousel, and many other types. Each one has a certain advantage and can be useful depending on the type of marketing you're doing.

One of the best ways to decide the type of ad to run is to look at what your competitors are doing.

Course completion certificate



greatlearning
Learning for Life

Certificate of completion

Presented to

Lavanya B

For successfully completing a free online course

Introduction to Digital Marketing

Provided by

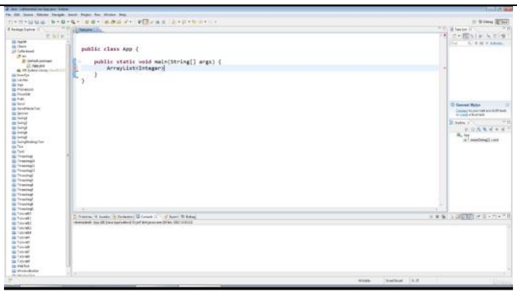
Great Learning Academy

(On June 2020)

To verify this certificate visit verify.greatlearning.in/OHEGEPDY

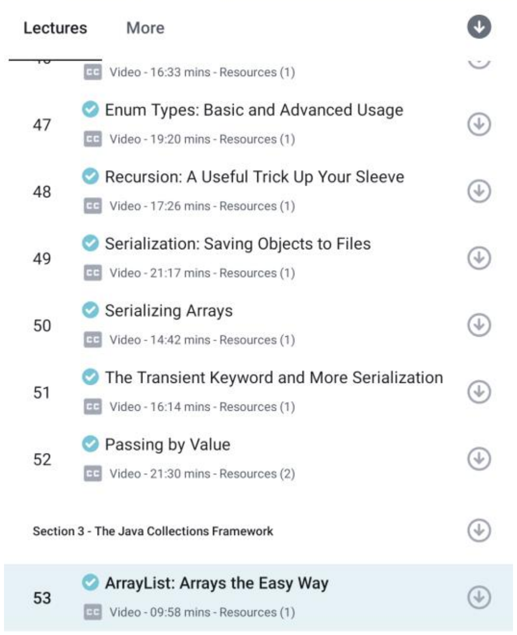
Date:	**15/06/2020**	**Name:**	**Lavanya B**
Course:	**Java**	**USN:**	**4al17ec043**
Topic:	**Programming core JAVA**	**Semester & Section:**	**6th A**

AFTERNOON SESSION DETAILS
Image of session



```
public class App {

    public static void main(String[] args) {
        ArrayList<Integer>
    }
}
```



Lectures More

Video - 16:33 mins - Resources (1)

47 Enum Types: Basic and Advanced Usage
Video - 19:20 mins - Resources (1)

48 Recursion: A Useful Trick Up Your Sleeve
Video - 17:26 mins - Resources (1)

49 Serialization: Saving Objects to Files
Video - 21:17 mins - Resources (1)

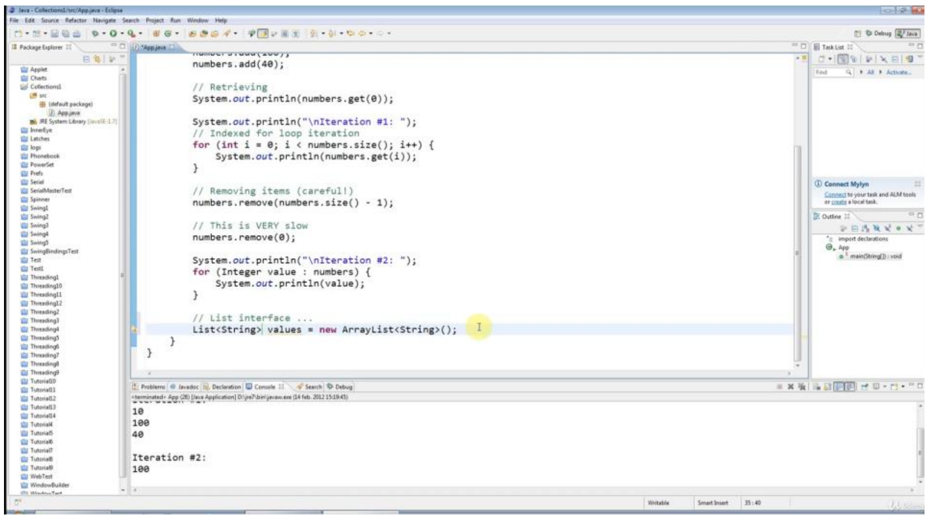
50 Serializing Arrays
Video - 14:42 mins - Resources (1)

51 The Transient Keyword and More Serialization
Video - 16:14 mins - Resources (1)

52 Passing by Value
Video - 21:30 mins - Resources (2)

Section 3 - The Java Collections Framework

53 ArrayList: Arrays the Easy Way
Video - 09:58 mins - Resources (1)



```
        numbers.add(40);

        // Retrieving
        System.out.println(numbers.get(0));

        System.out.println("\nIteration #1: ");
        // Indexed for loop iteration
        for (int i = 0; i < numbers.size(); i++) {
            System.out.println(numbers.get(i));
        }

        // Removing items (careful!)
        numbers.remove(numbers.size() - 1);

        // This is VERY slow
        numbers.remove(0);

        System.out.println("\nIteration #2: ");
        for (Integer value : numbers) {
            System.out.println(value);
        }

        // List interface ...
        List<String> values = new ArrayList<String>();
    }
}
```

```
<terminated> App (28) [Java Application] D:\jre7\bin\javaw.exe (14 feb. 2012 15:19:45)
10
100
40

Iteration #2:
100
```

Report

Enums

An Enum is a special type used to define collections of constants. Use Enums when a variable can only take one out of a small set of possible values.
If you use Enums instead of integers, you increase compile-time checking and avoid errors from passing in invalid constants, and you document which values are legal to use.

Eg:
```
public class Program {
   enum Rank {
      SOLDIER,
      SERGEANT,
      CAPTAIN
   }
   public static void main(String[] args) {
      Rank a = Rank.SOLDIER;

      switch(a) {
         case SOLDIER:
            System.out.println("Soldier says hi!");
            break;
         case SERGEANT:
            System.out.println("Sergeant says Hello!");
            break;
         case CAPTAIN:
            System.out.println("Captain says Welcome!");
            break;
      }
   }
}
```

Exceptions

An exception is a problem that occurs during program execution. Exceptions cause abnormal termination of the program.
Exception handling is a powerful mechanism that handles runtime errors to maintain normal application flow.

An exception can occur for many different reasons. Some examples:
- A user has entered invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications.
- Insufficient memory and other issues related to physical resources.

Eg:

```
public class MyClass {
   public static void main(String[ ] args) {
      try {
         int a[ ] = new int[2];
         System.out.println(a[5]);
      } catch (Exception e) {
         System.out.println("An error occurred");
      }
   }
}
```

Throws

The throw keyword allows you to manually generate exceptions from your methods. Some of the numerous available exception types include the IndexOutOfBoundsException, IllegalArgumentException, ArithmeticException. We can throw an ArithmeticException in our method when the parameter is 0.

The throws statement in the method definition defines the type of Exception(s) the method can throw.
Next, the throw keyword throws the corresponding exception, along with a custom message.
If we call the div method with the second parameter equal to 0, it will throw an ArithmeticException with the message "Division by Zero".

Eg:

```
public class Program {

   static int div(int a, int b) throws ArithmeticException {
      if(b == 0) {
         throw new ArithmeticException("Division by Zero");
      } else {
         return a / b;
      }
   }

   public static void main(String[] args) {
      System.out.println(div(42, 0));
   }

}
```

Threads

Java is a multi-threaded programming language. This means that our program can make optimal use of available resources by running two or more components concurrently, with each component handling a different task.
You can subdivide specific operations within a single application into individual threads that all run in parallel.

Creating Threads-

1. **Extend the Thread class: Inherit from the Thread class, override its run() method, and write the functionality of the thread in the run() method. Then you create a new object of your class and call it's start method to run the thread.**

   Eg:
   ```
   class Loader extends Thread {
      public void run() {
         System.out.println("Hello");
      }
   }

   class MyClass {
      public static void main(String[ ] args) {
         Loader obj = new Loader();
         obj.start();
      }
   }
   ```

2. **Implementing the Runnable interface: Implement the run() method. Then, create a new Thread object, pass the Runnable class to its constructor, and start the Thread by calling the start() method.**

   Eg:
   ```
   class Loader implements Runnable {
      public void run() {
         System.out.println("Hello");
      }
   }

   class MyClass {
      public static void main(String[ ] args) {
         Thread t = new Thread(new Loader());
         t.start();
      }
   }
   ```