1. What is BDD framework?

Ans. BDD framework i.e. Behavior Driven Development is a software development approach that allows the tester/business analyst to create test cases in simple text language (English).

The simple language used in the scenarios helps even non-technical team members to understand what is going on in the software project. This helps and improves communication among technical and non-technical teams, managers, and stakeholders.

## What Is BDD Behavior Driven Development?

BDD transpires from the TDD i.e. Test Driven Development which allows the users to work with multiple test data with minimum intervention in the software code and thereby helps to increase the reusability of the code, which is a time-saving mechanism in software development/ test automation.

**By inheriting TDD, BDD also has all those features along with its advantages.**

* Test scenarios are written separately in a different file, named as Feature file.
* Tests are written by focusing user stories and system behavior in a layman language.
* Code is subject to be written differently in step definitions file i.e. Java, Python.

### **Why Use BDD Framework?**

Before the BDD framework, everyone was using TDD. TDD works fine in software development, provided the stakeholders are familiar with the framework being used and their technical knowledge is sufficient. However, this may not be the case always.

BDD provides a path that acts as a bridge to overcome the gap between the technical and the non-technical teams because the test cases are commonly written in simple text, i.e. English. The main advantage of BDD is the low jargon and clearer approach which is easier to understand.

## Advantages Of BDD Framework

Enlisted below are the various advantages of BDD.

### **#1) Coverage of User Stories**

Hybrid Framework with BDD is meant to be combined with different features. Every resource in the software development phase can contribute to the BDD framework.

Due to its easy concept of layman text in the form of feature file allows the stakeholders of technical resources to write the scenarios in Gherkin language using the user stories. The compatibility of the plain text helps to gain maximum coverage on testing.

**Feature file containing scenarios are:**

* Defined user stories from the business.
* Criteria for the developers to determine if specifications meet business requirements.
* Test scenarios for the testing team.
* Shell cover for an automation tester which allows them to separately write their code in step definition files.
* Explained test scenarios for Stakeholders.

The classification of the step definitions helps the automation tester to keep his code untouched which thereby helps in the maintenance of the scripts.

### **#2) Clarity of Scenarios**

Gherkin language uses plain layman text that is focused on the outcome of the product which is being tested/developed using BDD.

As feature file separate the technical description in a different step definitions file for automation testers, it smartly helps a non-technical person to understand the automated test easily. Any updates can be implemented in a small discussion.

Readability power of gherkin guarantees the clarity of scenarios to each of its user which in turn, helps in building the right product.

### **#3) Automation of Test Scenarios**

Cucumber implementation in a BDD framework allows an automation tester to easily initiate the scripting with the right approach. Easy language of cucumber scenarios helps them to understand the functionality in a better way.

Cucumber is a language-independent plugin as it is compatible with many programming languages **E.g.** [Java](https://www.softwaretestinghelp.com/java/), [Python](https://www.softwaretestinghelp.com/python/), etc.

**Also Read =>**[**Automation Testing Using BDD Tool**](https://www.softwaretestinghelp.com/cucumber-bdd-tool-selenium-tutorial-30/)

### **#4) Code Reuse in Framework**

Given – When – Then approach gives liberty to the testers to use the same steps as many times we want in the feature file which gradually helps in saving time for the automation testers.

**Example:**

**Scenario: Scenario 1**

**Given**User is navigated to Google Home Page  
**When**User searched “Cucumber” in the search engine  
**Then**Clicked on the Search Button  
**And**User can see search results for Cucumber in the web browser

**Scenario: Scenario 2**

**Given**User is navigated to Google Home Page  
**When**User searched “Selenium” in the search engine  
**Then**Clicked on the Search Button  
**And**User can see search results for Selenium in the web browser

In the above two scenarios, we can conclude that “Given”, “When” and “Then” steps are reusable in the second scenario.

### **#5) Parameterization in Feature File**

A user can parameterize the gherkin steps in the feature file to obtain reusability in the file.

**For Example,** if a user is working on a bank application where he logs in to the application again and again. Such kind of steps could be parameterized with a different set of data and it saves time for the tester.

While writing the scenarios, the user has to define the feature file steps in such a way, so that the user can use the common functionality easily.

### **#6) Continuous Integration – Easy to Integrate**

Cucumber also supports working with Jenkins. You can run the cucumber test execution in Jenkins and also implement the same in Jenkins slave machines. The cucumber reporting plugin also provides users with an expanded view to track test scenarios.

1. **How to work on Data Driven Framework in Selenium Using Apache POI?**

Data Driven Framework is one of the popular Automation Testing Framework in the current market. Data Driven automated testing is a method in which the test data set is created in the excel sheet, and is then imported into automation testing tools to feed to the software under test.

Selenium Webdriver is a great tool to automate web-based applications. But it does not support read and write operations on excel files.

Therefore, we use [third party APIs](https://poi.apache.org/) like Apache POI.

### **What is Apache POI?**

Apache POI (Poor Obfuscation Implementation) is an API written in Java to support read and write operations – modifying office files. This is the most common API used for [Selenium data driven tests](https://www.softwaretestinghelp.com/data-driven-testing/).

### **Why data drive tests?**

Often there might be may be a number of data sets that have to be used to test a feature of an application. Now running the same test with different data manually is time-consuming, error prone and a boring task.

**Let us understand this scenario with an example.**

Suppose we need to test the login/Register/ Any form with multiple input fields with 100 different data sets.

**To test this you have three different approaches:**

**1)** Create 100 scripts one for each dataset and execute each test one by one.  
**2)** Change the data in the script and execute it multiple times.  
**3)** Import the data from the excel sheet and execute the script multiple times with different data.

First two scenarios are laborious, time-consuming – implying low ROI. Hence, we must follow the third approach.

In the third approach, are implementing the Data Driven framework, where all our data resides in an excel sheet, where it is imported from and used to test the features of the application.

### **What do we need to implement Data Driven Framework?**

In order to follow this approach we must have Eclipse, TestNG properly configured.

***Once done, we will look at:***

* Various interfaces of Apache POI.
* Integration of Apache POI in the Eclipse.
* Read Data from the Excel Sheet.
* Write data to the Excel Sheet.
* Advantages of using Apache POI with Selenium.

### **Interface in POI**

One of the most remarkable features of **Apache POI** is that it supports read and write operations on both .xls and .xslx files.

Below mentioned are some of the **interfaces of POI**.

* **XSSFWorkbook:** Represents workbook in xlsx file.
* **HSSFWorkbook:** Represents workbook in xls file.
* **XSSFSheet:** Represents a sheet in XLSX file.
* **HSSFSheet:** Represents a sheet in XLS file.
* **XSSFRow:** Represents a row in a sheet of XLSX file.
* **HSSFRow:** Represents a row in a sheet of XLS file.
* **XSSFCell:** Represents a cell in a row of XLSX file.
* **HSSFCell:** Represents a cell in a row of XLS file.

**Fields available in a cell:**

* **CELL\_TYPE\_BLANK:** Represents a blank cell.
* **CELL\_TYPE\_BOOLEAN:** Represents a Boolean cell (true or false).
* **CELL\_TYPE\_ERROR:** Represents an error value in a cell.
* **CELL\_TYPE\_FORMULA:** Represents a formula result on a cell.
* **CELL\_TYPE\_NUMERIC:** Represents numeric data in a cell.
* **CELL\_TYPE\_STRING:** Represents string in a cell.

### **The steps to use Selenium with Apache POI**

Let us create an automation script to test the login process of a web -based applications.

**Here, I have taken LinkedIn** **as an example**.

We import data from an excel sheet and then use it to log into the application and after execution, we write the result in the excel sheet.

**We need the following software installed on our system to carry on with the steps to execute the framework:**

* Java JDK 1.7+
* Eclipse IDE
* TestNG
* Selenium jars
* Microsoft Office / Open Office

**Step #1)**

Firstly, we need to configure Eclipse with **Apache POI**.

[Download](https://archive.apache.org/dist/poi/release/bin/poi-bin-3.10-FINAL-20140208.zip) jar files for Apache POI.

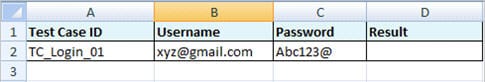
**Step #2)**

Unzip the jar file, and add the following jars to your project and configure them.

* dom4j-1.6.1.jar
* poi-3.10-FINAL-20140208.jar
* poi-ooxml-3.10-FINAL-20140208.jar
* poi-ooxml-schemas-3.10-FINAL-20140208.jar
* xmlbeans-2.3.0.jar

**Step #3)**

After configuring the respective jars, create an excel sheet enter some data in it and save it as TestData.xlsx at your preferred location.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/04/excel-sheet.jpg)

**Step #4)**

Now let us follow the sample code to read data from the excel sheet and use it to login to [linkedin.com](https://www.linkedin.com/).

|  |
| --- |
| package automationFramework;    import java.io.File;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.util.concurrent.TimeUnit;    import org.apache.poi.hssf.usermodel.HSSFCell;  import org.apache.poi.hssf.usermodel.HSSFSheet;  import org.apache.poi.hssf.usermodel.HSSFWorkbook;  import org.apache.poi.ss.usermodel.Cell;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  import org.openqa.selenium.support.ui.WebDriverWait;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;    /\*\*   \* @author Admin   \*   \*/  public class ReadWriteExcel  {      WebDriver driver;      WebDriverWait wait;      HSSFWorkbook workbook;      HSSFSheet sheet;      HSSFCell cell;     @BeforeTest   public void TestSetup()  {      // Set the path of the Firefox driver.      System.setProperty("webdriver.gecko.driver", "C:\\Users\\geckodriver.exe");      driver = new FirefoxDriver();        // Enter url.      driver.get("http://www.linkedin.com/");      driver.manage().window().maximize();        wait = new WebDriverWait(driver,30);      driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);  }     @Test   public void ReadData() throws IOException   {       // Import excel sheet.       File src=new File("C:\\Users\\Admin\\Desktop\\TestData.xls");         // Load the file.       FileInputStream finput = new FileInputStream(src);         // Load he workbook.      workbook = new HSSFWorkbook(finput);         // Load the sheet in which data is stored.       sheet= workbook.getSheetAt(0);         for(int i=1; i&lt;=sheet.getLastRowNum(); i++)       {           // Import data for Email.           cell = sheet.getRow(i).getCell(1);           cell.setCellType(Cell.CELL\_TYPE\_STRING);           driver.findElement(By.id("login-email")).sendKeys(cell.getStringCellValue());             // Import data for password.           cell = sheet.getRow(i).getCell(2);           cell.setCellType(Cell.CELL\_TYPE\_STRING);           driver.findElement(By.id("login-password")).sendKeys(cell.getStringCellValue());            }    }    } |

**Step #5)**

Right click on the test case class and click on Run as –> TestNG Test.

**Apache POI** imports data from the excel sheet and uses it to log into our application. Now that we saw how to read data from the excel sheet, let’s look at how to write to the sheet.

|  |
| --- |
| package automationFramework;    import java.io.File;  import java.io.FileInputStream;  import java.io.FileOutputStream;  import java.io.IOException;  import java.util.concurrent.TimeUnit;    import org.apache.poi.hssf.usermodel.HSSFCell;  import org.apache.poi.hssf.usermodel.HSSFSheet;  import org.apache.poi.hssf.usermodel.HSSFWorkbook;  import org.apache.poi.ss.usermodel.Cell;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  import org.openqa.selenium.support.ui.WebDriverWait;  import org.testng.annotations.BeforeTest;  import org.testng.annotations.Test;    /\*\*   \* @author Admin   \*   \*/  public class ReadWriteExcel  {      WebDriver driver;      WebDriverWait wait;      HSSFWorkbook workbook;      HSSFSheet sheet;      HSSFCell cell;     @BeforeTest   public void TestSetup()  {      // Set the path of the Firefox driver.      System.setProperty("webdriver.gecko.driver", "C:\\Users\\geckodriver.exe");      driver = new FirefoxDriver();        // Enter url.      driver.get("http://www.linkedin.com/");      driver.manage().window().maximize();        wait = new WebDriverWait(driver,30);      driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);  }     @Test   public void ReadData() throws IOException   {       // Import excel sheet.       File src=new File("C:\\Users\\Admin\\Desktop\\TestData.xls");         // Load the file.       FileInputStream finput = new FileInputStream(src);         // Load he workbook.      workbook = new HSSFWorkbook(finput);         // Load the sheet in which data is stored.       sheet= workbook.getSheetAt(0);         for(int i=1; i&lt;=sheet.getLastRowNum(); i++)       {           // Import data for Email.           cell = sheet.getRow(i).getCell(1);           cell.setCellType(Cell.CELL\_TYPE\_STRING);           driver.findElement(By.id("login-email")).sendKeys(cell.getStringCellValue());             // Import data for password.           cell = sheet.getRow(i).getCell(2);           cell.setCellType(Cell.CELL\_TYPE\_STRING);           driver.findElement(By.id("login-password")).sendKeys(cell.getStringCellValue());             // Write data in the excel.         FileOutputStream foutput=new FileOutputStream(src);            // Specify the message needs to be written.          String message = "Data Imported Successfully.";            // Create cell where data needs to be written.          sheet.getRow(i).createCell(3).setCellValue(message);            // Specify the file in which data needs to be written.          FileOutputStream fileOutput = new FileOutputStream(src);            // finally write content          workbook.write(fileOutput);             // close the file          fileOutput.close();         }   }  } |

**Note:** If you encounter any problems during this process, please check the following points.

* Make sure all the mentioned jars are added to the project and are properly configured.
* Required software is correctly installed.
* Proper use of an interface with respect to excel file, like HSSF for .xls and XSSF for .xlsx.
* Valid row and column index is used.
* Excel file must be closed before execution.
* Proper classes used for the excel file like XSSF used for .xlsx files and HSSF used for .xls files.

### **Advantages of using Data Driven Framework**

* Improves test coverage.
* Re-usability of code.
* Less maintenance.
* Faster Execution.
* Permits better error handling.

### **What Is Defect Severity And Priority?**

Priority by the English definition is used in the comparison of two things or conditions, where one has to be given more importance than the other(s) and has to be tackled with/resolved first before proceeding to the next one(s). Therefore in the context of defects, the priority of a defect would indicate the urgency with which it would need to be fixed.

Severity by the English definition is used to describe the gravity of an undesirable occurrence. Hence when it comes to bugs, the severity of a bug would indicate the effect it has on the system in terms of its impact.

#### **Who Defines These?**

QA classifies the defect under appropriate severity based on the complexity and criticality of the defects.

Any business stakeholders including the project managers, business analysts, product owner define the priority of the defects.

#### **Priority & Severity Scenarios**

#### **#1) High Severity and High Priority**

Any Critical/major business case failure automatically gets promoted to this category.

Any defects due to which the testing cannot continue at any cost or causes a severe system failure to fall into this category. **For Example,**clicking on a particular button doesn't load the feature itself. Or performing a particular function brings down the server consistently and causes data loss. The red lines in the above figure indicate these kinds of defects.

**For Example,**

The system crashes after you made the payment or when you are not able to add the items to the Cart, this defect is marked as High Severity and High Priority defect.

**Another example** would be ATM vending currency feature wherein after entering the correct username and the password, the machine does not dispense money but deducts the transferred from your account.

#### **#2) High Priority and Low Severity**

Any minor severity defects that could directly impact the user experience automatically gets promoted to this category.

Defects that have to be fixed but do not affect the application come under this category.

**For Example,** the feature is expected to display a particular error to the user with respect to its return code. In this case, functionally the code will throw an error, but the message will need to be more relevant to the return code generated. The blue lines in the figure indicate these kinds of defects.

**For Example,**

The logo of the company in the front-page is wrong, it is considered to be **High Priority and Low Severity**defect**.**

**Example 1)** In the Online shopping website when the FrontPage logo is spelled wrong, for example instead of Flipkart it is spelled as Flipkart.

**Example 2)** In the bank logo, instead of ICICI, it is written as ICCCI.

In terms of functionality, it is not affecting anything so we can mark as Low Severity, but it has an impact on user experience. This kind of defect needs to be fixed on high priority even though they have very less impact on the application side.

#### **#3) High Severity and Low Priority**

Any defect that is functionally not meeting the requirements or have any functional implications on the system but sidelined to back seat by the stakeholders when it comes to business criticality automatically gets promoted to this category.

Defects that have to be fixed but not immediately. This can specifically occur during ad-hoc testing. It means that the functionality is affected to a large extent, but is observed only when certain uncommon input parameters are used.

**For Example,** a particular functionality can be used only on a later version of the firmware, so in order to verify this – the tester actually downgrades his system and performs the test and observes a serious functionality issue that is valid. In such a case the defects will be classified in this category denoted by pink lines, as normally end users will be expected to have a higher version of the firmware.

**For Example,**

In a social networking site, if a beta version of a new feature is released with not many active users using that facility as of today. Any defect found on this feature can be classified as a low priority as the feature takes back seat due to business classification as not important.

Though this feature is having a functional defect, as it is not impacting the end customers directly, a business stakeholder can classify the defect under low priority though it has a severe functional impact on the application.

This is a high severity fault but can be prioritized to low priority as it can be fixed with the next release as a change request. Business stakeholders also prioritize this feature as a rarely used feature and do not impact any other features that have a direct impact on user experience. This kind of defect can be classified under the **High Severity but Low Priority** category.

#### **#4) Low Severity and Low Priority**

Any spelling mistakes /font casing/ misalignment in the paragraph of the 3rd or 4th page of the application and not in the main or front page/ title.

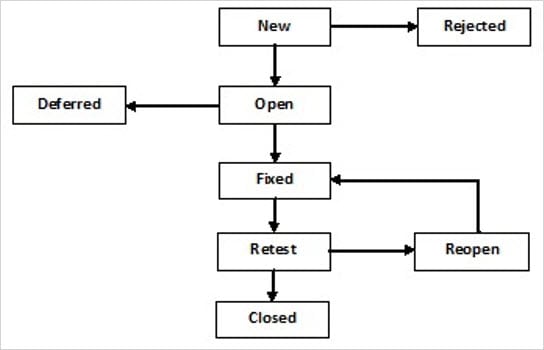
These defects are classified in the green lines as shown in the figure and occur when there is no functionality impact, but still not meeting the standards to a small degree. Generally cosmetic errors or say dimensions of a cell in a table on UI are classified here.

1. Defect Life cycle:

### A Defect life cycle, also known as a Bug life cycle, is a cycle of a defect from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won’t get reproduced again.

### **Defect Workflow**

It is now time to understand the actual workflow of a Defect Life Cycle with the help of a simple diagram as shown below.

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Defect-Life-cycle-In-HPM.jpg)

### **Defect States**

**#1) New**: This is the first state of a defect in the Defect Life Cycle. When any new defect is found, it falls in a ‘New’ state, and validations and testing are performed on this defect in the later stages of the Defect Life Cycle.

**#2) Assigned:** In this stage, a newly created defect is assigned to the development team for working on the defect. This is assigned by the project lead or the manager of the testing team to a developer.

**#3) Open:**Here, the developer starts the process of analyzing the defect and works on fixing it, if required. If the developer feels that the defect is not appropriate then it may get transferred to any of the below four states namely **Duplicate, Deferred, Rejected, or Not a Bug**-based upon the specific reason.

I will discuss these four states in a while.

**#4) Fixed:**When the developer finishes the task of fixing a defect by making the required changes then he can mark the status of the defect as ‘Fixed’.

**#5) Pending Retest:**After fixing the defect, the developer assigns the defect to the tester for retesting the defect at their end, and till the tester works on retesting the defect, the state of the defect remains in ‘Pending Retest’.

**#6) Retest:**At this point, the tester starts the task of working on the retesting of the defect to verify if the defect is fixed accurately by the developer as per the requirements or not.

**#7) Reopen:**If any issue persists in the defect then it will be assigned to the developer again for testing and the status of the defect gets changed to ‘Reopen’.

**#8) Verified:**If the tester does not find any issue in the defect after being assigned to the developer for retesting and he feels that if the defect has been fixed accurately then the status of the defect gets assigned to ‘Verified’.

**#9) Closed:**When the defect does not exist any longer then the tester changes the status of the defect to ‘Closed’.

**Few More:**

* **Rejected:**If the defect is not considered as a genuine defect by the developer then it is marked as ‘Rejected’ by the developer.
* **Duplicate:**If the developer finds the defect as same as any other defect or if the concept of the defect matches any other defect then the status of the defect is changed to ‘Duplicate’ by the developer.
* **Deferred:**If the developer feels that the defect is not of very important priority and it can get fixed in the next releases or so in such a case, he can change the status of the defect as ‘Deferred’.
* **Not a Bug:**If the defect does not have an impact on the functionality of the application then the status of the defect gets changed to ‘Not a Bug’.

1. **What type of defect is a not reproducible defect?**

**Answer:** A defect that is not occurring repeatedly in every execution and is producing only at some instances and whose steps as proof have to be captured with the help of screenshots, then such a defect is called as a ‘not reproducible’ defect.

1. **What details are included in a defect report?**

**Answer:** **A defect report consists of the following details:**

Defect ID, Description of the defect, Feature Name, Test Case Name, Reproducible defect or not, Status of a defect, Severity, and Priority of a defect, Tester Name, Date of testing the defect, Build Version in which the defect was found.

And the Developer to whom defect has been assigned, name of the person who has fixed the defect, Screenshots of a defect depicting the flow of the steps, Fixing the date of a defect, and the person who has approved the defect.

1. What is encapsulation in java? Why we use it?

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates.Other way to think about encapsulation is, it is a protective shield that prevents the data from being accessed by the code outside this shield.

* Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of own class in which they are declared.
* As in encapsulation, the data in a class is hidden from other classes using the data hiding concept which is achieved by making the members or methods of class as private and the class is exposed to the end user or the world without providing any details behind implementation using the abstraction concept, so it is also known as **combination of data-hiding and abstraction.**.
* Encapsulation can be achieved by: Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

// Java program to demonstrate encapsulation

public class Encapsulate

{

// private variables declared

// these can only be accessed by

// public methods of class

private String geekName;

private int geekRoll;

private int geekAge;

// get method for age to access

// private variable geekAge

public int getAge()

{

return geekAge;

}

// get method for name to access

// private variable geekName

public String getName()

{

return geekName;

}

// get method for roll to access

// private variable geekRoll

public int getRoll()

{

return geekRoll;

}

// set method for age to access

// private variable geekage

public void setAge( int newAge)

{

geekAge = newAge;

}

// set method for name to access

// private variable geekName

public void setName(String newName)

{

geekName = newName;

}

// set method for roll to access

// private variable geekRoll

public void setRoll( int newRoll)

{

geekRoll = newRoll;

}

}

**Advantages of Encapsulation**:

* **Data Hiding:** The user will have no idea about the inner implementation of the class. It will not be visible to the user that how the class is storing values in the variables. He only knows that we are passing the values to a setter method and variables are getting initialized with that value.
* **Increased Flexibility:** We can make the variables of the class as read-only or write-only depending on our requirement. If we wish to make the variables as read-only then we have to omit the setter methods like setName(), setAge() etc. from the above program or if we wish to make the variables as write-only then we have to omit the get methods like getName(), getAge() etc. from the above program
* **Reusability:** Encapsulation also improves the re-usability and easy to change with new requirements.
* **Testing code is easy:** Encapsulated code is easy to test for unit testing.

This is the process of hiding information. The other objects cannot access the data directly. Instead, they have to invoke the getters which are designed to protect the data from misuse or unwanted changes.

What if we want the outside world to be able to modify our data in a safety manner? In this case, we can provide methods in the pattern of setXXX() - the so called **setters** in Java. For example, creating a setter for the field name:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | public void setName(String name) {      if (name == null || name.equals("")) {          throw new IllegalArgumentException("name cannot be null or empty!");      }        this.name = name;  } |

Here, someone can change the name of a Person object via this setter method, but he cannot set an empty or null name - as the setter will throw an exception if doing so. This protects data from misuse or malicious changes. For example:

|  |  |
| --- | --- |
| 1  2  3  4  5 | Person p = new Person();    p.setName("");  // ERROR: IllegalArgumentException will be thrown    p.setName("Tom");   // OK, legal |

Likewise, we can implement a setter for the age attribute that allows the caller to set age in a valid range. For example:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | public void setAge(int age) {      if (age < 18 || age > 55) {          throw IllegalArgumentException("Age must be from 18 to 55");      }      this.age = age;  } |

So far you’ve got an understanding about what encapsulation is in OOP and how it is implemented in Java. In short, encapsulation is a technique for hiding implementation details and restricting access for the outside world.

1. Difference between Abstract class and Interface?

* Main difference is methods of a Java interface are implicitly abstract and cannot have implementations. A Java abstract class can have instance methods that implements a default behavior.
* Variables declared in a Java interface is by default final. An  abstract class may contain non-final variables.
* Members of a Java interface are public by default. A Java abstract class can have the usual flavors of class members like private, protected, etc..
* Java interface should be implemented using keyword “implements”; A Java abstract class should be extended using keyword “extends”.
* An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.
* A Java class can implement multiple interfaces but it can extend only one abstract class.
* Interface is absolutely abstract and cannot be instantiated; A Java abstract class also cannot be instantiated, but can be invoked if a main() exists.
* In comparison with java abstract classes, java interfaces are slow as it requires extra indirection.

1. What is product backlog?

A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.

The product backlog is the single authoritative source for things that a team works on.

1. Backlog Refinement?

Backlog refinement (formerly known as backlog grooming) is when the [product owner](https://www.agilealliance.org/glossary/product-owner/)and some, or all, of the rest of the team review items on the [backlog](https://www.agilealliance.org/glossary/backlog/) to ensure the backlog contains the appropriate items, that they are prioritized, and that the items at the top of the backlog are ready for delivery. This activity occurs on a regular basis and may be an officially scheduled meeting or an ongoing activity. Some of the activities that occur during this refinement of the backlog include:

* removing [user stories](https://www.agilealliance.org/glossary/user-stories/) that no longer appear relevant
* creating new user stories in response to newly discovered needs
* re-assessing the relative priority of stories
* assigning estimates to stories which have yet to receive one
* correcting estimates in light of newly discovered information
* [splitting](https://www.agilealliance.org/glossary/split/) user stories which are high priority but too coarse grained to fit in an upcoming [iteration](https://www.agilealliance.org/glossary/iteration/)

1. Who closes the bug ?

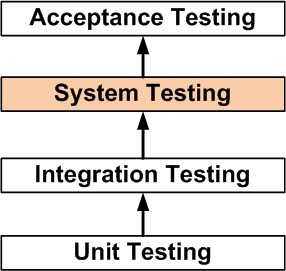
Tester

1. Regression Testing & Retesting?

|  |  |
| --- | --- |
| **Regression testing** | **Retesting** |
| Regression testing is done to find out the issues which may get introduced because of any change or modification in the application. | Retesting is done to confirm whether the failed test cases in the final execution are working fine or not after the issues have been fixed. |
| The purpose of regression testing is that any new change in the application should NOT introduce any new bug in existing functionality. | The purpose of retesting is to ensure that the particular [**bug**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/)or issue is resolved and the functionality is working as expected. |
| [**Verification**](http://tryqa.com/what-is-verification-in-software-testing-or-what-is-software-verification/)of bugs are not included in the regression testing. | Verification of bugs are included in the retesting. |
| Regression testing can be done in parallel with retesting. | Retesting is of high [**priority**](http://tryqa.com/what-is-the-difference-between-severity-and-priority/)so it’s done before the regression testing. |

1. System Testing?

**SYSTEM TESTING**is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements.



Definition by ISTQB

* **system testing:**The process of testing an integrated system to verify that it meets specified requirements.

System Testing is the third [level of software testing](https://softwaretestingfundamentals.com/software-testing-levels/) performed after [Integration Testing](https://softwaretestingfundamentals.com/integration-testing/) and before [Acceptance Testing](https://softwaretestingfundamentals.com/acceptance-testing/).

1. Can we have a private constructor?

Yes, class can have a private constructor. It is needed as to disallow to access the constructor from other classes and remain it accessible within defined class.

A singleton is a design pattern that allows only one instance of your class to be created, and this can be accomplished by using a private constructor.

## How To Select Correct Test Cases For Automation Testing?

### 1. Repetitive Test Runs

* This is the golden rule. If the test run is repetitive, you should automate it. This will save your precious time and energy. In addition, if this is the test you assume that you will run frequently in the future, automating is a very good idea. You will have more accurate and precise results, each time.

### 2. High-Risk Test Cases

* If the involved risk is high, you should automate that test case. Prioritize the test cases based on human error hazard potential.

### 3. Critical Parts of A Web Application

* Automating critical parts of your web app is a smart idea. You want to avoid any possibility of man-made error and to be on the safe side. This is especially true for test cases that can jeopardize the whole web application.

### 4. Extensive Tests

* Inputting lots of data can be exhausting. In addition, the chances of making a mistake during the manual testing increase. This is why test which implies large data sets are usually automated. Mainly, we refer to those test cases which require filling long forms.

### 5. Evaluate the Pros and Cons

* By evaluation, we imply the time required to complete the automation process or to do the test manually. If you are not ready to invest time in the automation process, i.e. if the process itself would take more time than manual testing, common sense says that the automation is not worth it. This is especially true for non-repetitive tests.

### 6. Three Golden Tests

* There are three test cases that you should not question when it comes to automation: the [regression test](https://www.lambdatest.com/blog/building-a-regression-testing-strategy-for-agile-teams/), smoke test, and sanity test. Why? Because each build or release of the app requires testing.
* To perform these tests, there are various tools available at your aid. You can utilize tools like [LambdaTest](https://www.lambdatest.com/blog/building-a-regression-testing-strategy-for-agile-teams/" \t "_blank)which can help you test across 2,000+ browsers and operating system combinations. You can perform both manual and automation testing using LambdaTest.

### 7. If You Can Answer the Magical Question “Why?”

* Don’t ever automate a test case just because you can. This could bring you more harm than good. If it will benefit the app, your team, or the organization, go ahead. Your goal should always be fast feedback, precision, and better overall results. However, don’t forget that Selenium requires deep knowledge when it comes to script writing.
* Manual testing is sometimes a better option, especially if you are not a skilled Selenium user.
* Remember that the answer to the question “Why?” when it comes to automation must never be “Because I can.”

### 8. Complex Cases

* Automating complex test cases is almost a rule. Again, you do want to be on the safe side and to save your time when it comes to complexity.

### 9. Performance Testing

* We usually choose to automate performance testing, such as load testing and stress testing.

### 10. Long Testing

* If a case requires an overnight devotion, you should automate it.

## How to get all values from a dropdown?

  To get the default values selected in dropdown

|  |
| --- |
| String s= driver.findElement(By.*xpath*("//select[@id='ddladult1']")).getText(); |
| System.***out***.println("Default value selected is :"+ s); |

  To get all the dropdown values

**Method 1**

|  |
| --- |
| Select select = **new**Select(driver.findElement(By.*xpath*("//select[@id='ddladult1']"))); |
| select.getOptions(); |

**Method 2**

|  |
| --- |
| WebElement dropdown = driver.findElement(By.*xpath*("//select[@id='ddladult1']"));          Select select = **new** Select(dropdown);         java.util.List<WebElement> options = select.getOptions();  **for**(WebElement item:options)          {                 System.out.println("Dropdown values are "+ item.getText());             } |

  To select a desired value.

|  |
| --- |
| Select dropdownValue= new Select(driver.findElement(By.xpath("("//select[@id='ddladult1']"))); |
| dropdownValue.selectByIndex(3); |

# **Automatically Re-Running Failed Scenarios in Cucumber- Junit**

Step 1: In Runner File inside plugin we need to write “rerun:rerun/failed\_scenarios.txt”. Cucumber will write the failed scenario and line number in the generated failed\_scenarios.txt file.

|  |  |
| --- | --- |
|  | import cucumber.api.CucumberOptions; |
|  | import cucumber.api.junit.Cucumber; |
|  | import org.junit.runner.RunWith; |
|  |  |
|  | @RunWith(Cucumber.class) |
|  | @CucumberOptions( |
|  | features = {"C:\\Users\\ankur.jain\\eclipse-workspace\\Cucumber\_Test\\src\\main\\java\\com\\qa\\FeatuerFile\\Login.feature", |
|  | }, //the path of the feature files |
|  | //format= {"html:target/site/cucumber-pretty;json:target/cucumber.json&quot"}, |
|  | glue={"com.qa.StepDefinition"}, //the path of the step definition files |
|  | plugin= {"pretty","html:target/site/cucmber-pretty", "json:target/cucumber/cucumber.json","rerun:rerun/failed\_scenarios.txt"}, //to generate different types of reporting |
|  | monochrome = true, //display the console output in a proper readable format |
|  | strict = true, //it will check if any step is not defined in step definition file |
|  | dryRun = false//to check the mapping is proper between feature file and step def file |
|  |  |
|  | //tags = {"@all"} |
|  | ) |
|  |  |
|  | public class TestRunner |
|  | { |
|  |  |
|  | } |

[**view raw**](https://gist.github.com/ankjain53/2c8b6ea41503cf19ae1d25b2a5e6839b/raw/48591bb53c251261f6547da13ff6da225d9ce784/TestRunner.java)[**TestRunner.java**](https://gist.github.com/ankjain53/2c8b6ea41503cf19ae1d25b2a5e6839b#file-testrunner-java) hosted with ❤ by [**GitHub**](https://github.com/)

Step 2: We Need to Create another Runner File. Let’s Name it as ReRunRunner.Java and inside feature we need to provide the name of this failed\_scenario.txt file along with @.

|  |  |
| --- | --- |
|  | import cucumber.api.CucumberOptions; |
|  | import cucumber.api.junit.Cucumber; |
|  | import org.junit.runner.RunWith; |
|  |  |
|  | @RunWith(Cucumber.class) |
|  | @CucumberOptions( |
|  | features = {"@rerun/failed\_scenarios.txt", |
|  | }, |
|  | //Cucumber execute failed scenarios from this file |
|  | //format= {"html:target/site/cucumber-pretty;json:target/cucumber.json&quot"}, |
|  | glue={"com.qa.StepDefinition"}, //the path of the step definition files |
|  | plugin= {"pretty","html:target/site/cucmber-pretty", "json:target/cucumber/cucumber.json", "junit:target/cucumber.xml","com.cucumber.listener.ExtentCucumberFormatter:target/html/ExtentReport.html"}, //to generate different types of reporting |
|  | monochrome = true, //display the console output in a proper readable format |
|  | strict = true, //it will check if any step is not defined in step definition file |
|  | dryRun = false//to check the mapping is proper between feature file and step def file |
|  | //tags = {"@all"} |
|  | ) |
|  |  |
|  | public class ReRunRunner |
|  | { |
|  | } |

## Difference between Regression & Sanity testing?

## 

## What is STLC?

Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In the STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; however, the basis remains the same.

**Below are the phases of STLC:**

1. Requirements phase
2. Planning Phase
3. Analysis phase
4. Design Phase
5. Implementation Phase
6. Execution Phase
7. Conclusion Phase
8. Closure Phase

## What are test cases for old mobile?

* Verify that all the required buttons- numbers 0-9, calling buttons etc are present
* Verify that the user can make a call by pressing numbers and hitting calling(green) button
* Verify that user can make a call by selecting a contact person from the phone directory
* Verify that the user can reject an incoming call
* Verify that the user can receive an SMS
* Verify that the user can type and send an SMS
* Verify that the dimension of the mobile is as per specification
* Verify the screen size of the mobile
* Verify that the weight of the mobile is as per the specification
* Verify the font type and size of the characters printed on the keypad
* Verify the color of the mobile phone’s outer body and characters printed on keypad
* Verify the pressure required to press a key on the keypad
* Verify that spacing between the keys on the keypad are adequate
* Check the type of mobile- smartphone or normal
* Check if the mobile is colored or black-white
* Check the lighting on the mobile screen is adequate- verify in dark daylight
* Check if a mobile phone can be locked out without password or pin
* Check if mobile phone can be locked out with password or pin
* Verify that the mobile phone can be unlocked with/without password
* Verify that the user can receive a call when the phone is locked
* Verify that receiving a call when phone is locked, doesn’t unlock it after call completion
* Verify that user can select an incoming call and SMS alert ringtone
* Verify that the user can make silent or vibrate mode or incoming calls and SMS
* Verify the battery requirement of the mobile
* Verify the total time taken to charge the mobile completely
* Verify the total time for mobile to get completely discharged when left idle
* Verify the total talk for mobile to get completely discharged when continuously used in conversation
* Verify the length of charger wire
* Verify that mobile can be switched off and ON
* Verify that user can store contact details on the phone book directory
* Verify that user can delete and update contact details in the phonebook directory
* Verify that Call logs are maintained in the Call Logs
* Verify that received and Sent SMSs are saved in mobile
* Verify that user can silent the phone during an incoming call
* Verify the auto-reject option can be applied and removed on particular numbers

1. What is smoke testing?

**SMOKE TESTING**, also known as “Build Verification Testing”, is a type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most important functions work. The result of this testing is used to decide if a build is stable enough to proceed with further testing.

# What Is Boundary Value Analysis And Equivalence Partitioning?

**Test cases for input box accepting numbers between 1 and 1000 using Equivalence Partitioning:**

**#1)** One input data class with all valid inputs. Pick a single value from range 1 to 1000 as a valid test case. If you select other values between 1 and 1000 the result is going to be the same. So one test case for valid input data should be sufficient.

**#2)** Input data class with all values below the lower limit. I.e. any value below 1, as an invalid input data test case.

**#3)** Input data with any value greater than 1000 to represent the third invalid input class.

So using Equivalence Partitioning you have categorized all possible test cases into three classes. Test cases with other values from any class should give you the same result.

We have selected one representative from every input class to design our test cases. Test case values are selected in such a way that largest number of attributes of equivalence class can be exercised.

Equivalence Partitioning uses fewest test cases to cover maximum requirements.

### **Boundary Value Analysis**

It's widely recognized that input values at the extreme ends of the input domain cause more errors in the system. More application **errors occur at the boundaries** of the input domain. ‘Boundary Value Analysis' Testing technique is used to identify errors at boundaries rather than finding those that exist in the center of the input domain.

Boundary Value Analysis is the next part of Equivalence Partitioning for designing test cases where test cases are selected at the edges of the equivalence classes.

**Test cases for input box accepting numbers between 1 and 1000 using Boundary value analysis:**

**#1)** Test cases with test data exactly as the input boundaries of input domain i.e. values 1 and 1000 in our case.

**#2)** Test data with values just below the extreme edges of input domains i.e. values 0 and 999.

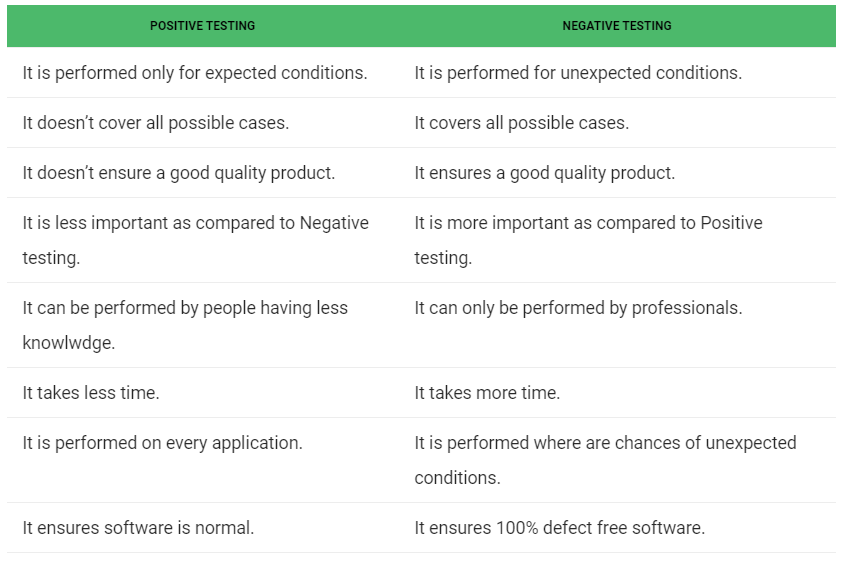
**#3)** Test data with values just above the extreme edges of the input domain i.e. values 2 and 1001.

Boundary Value Analysis is often called as a part of the Stress and Negative Testing.

1. What is acceptance testing?

**acceptance testing:**Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

1. What is positive and negative testing?



## What is test strategy and test plan?

## A Test Strategy document is a high level document and normally developed by project manager. This document defines “Software Testing Approach” to achieve testing objectives.

The Test Strategy is normally derived from the Business Requirement Specification document.

The Test Strategy document is a static document meaning that it is not updated too often.

### Components of the Test Strategy document

* Scope and Objectives
* Business issues
* Roles and responsibilities
* Communication and status reporting
* Test deliverables
* Industry standards to follow
* Test automation and tools
* Testing measurements and metrices
* Risks and mitigation
* Defect reporting and tracking
* Change and configuration management
* Training plan

## Test Plan

The Test Plan document on the other hand, is derived from the Product Description, Software Requirement Specification SRS, or Use Case Documents.  
The Test Plan document is usually prepared by the Test Lead or Test Manager and the focus of the document is to describe what to test, how to test, when to test and who will do what test.

Components of the Test Plan document

Test Plan id

Introduction

Test items

Features to be tested

Features not to be tested

Test techniques

Testing tasks

Suspension criteria

Features pass or fail criteria

Test environment (Entry criteria, Exit criteria)

Test deliverables

Staff and training needs

Responsibilities

Schedule

1. Test closure documents?

**Test Closure** is a **document** that gives a summary of all the **tests** conducted during the software development life cycle, it also gives a detailed analysis of the bugs removed and errors found . In other words, **Test Closure** is a memo that is prepared prior to formally completing the **testing** process

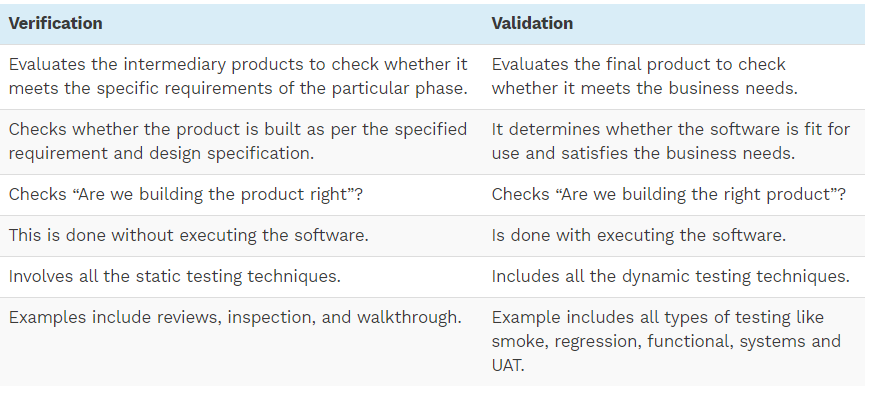
1. What is integration testing?

**NTEGRATION TESTING** is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

1. What is monkey testing(ADHOC Testing)?

**Monkey testing** is a technique in software **testing** where the user **tests** the application by providing random inputs and checking the behavior (or try to crash the application). Mostly this technique is done automatically where the user enters any random invalid inputs and checks the behavior

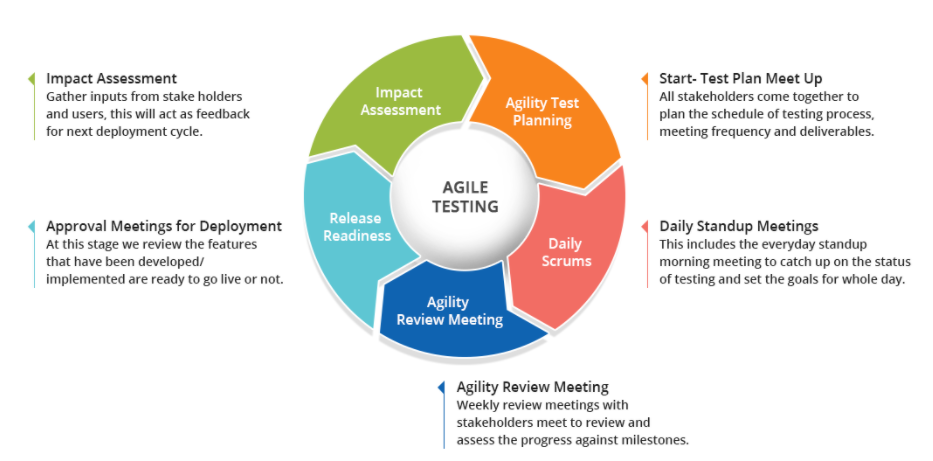
1. What is difference between Validation & Verification?



1. What is agile methodology in testing?

The agile testing life cycle includes the following 5 phases:

1. Impact assessment
2. Agile Testing Planning
3. Release Readiness
4. Daily Scrums
5. Test Agility Review



## What is abstraction in java?

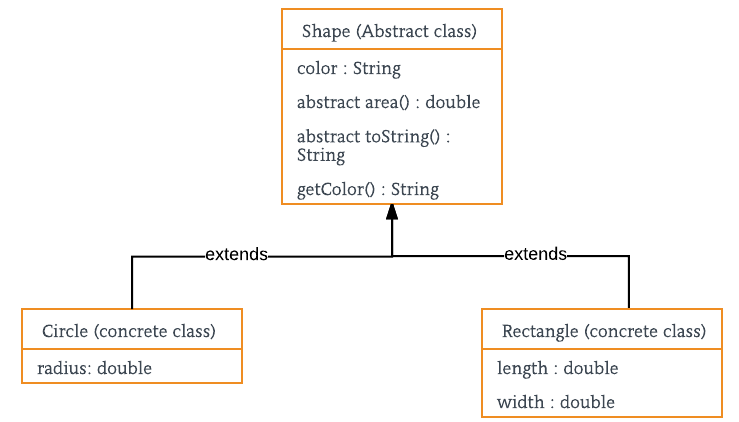
## Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details.The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

1. An abstract class is a class that is declared with [abstract keyword.](https://www.geeksforgeeks.org/abstract-keyword-in-java/)
2. An abstract method is a method that is declared without an implementation.
3. An abstract class may or may not have all abstract methods. Some of them can be concrete methods
4. A method defined abstract must always be redefined in the subclass,thus making [overriding](http://contribute.geeksforgeeks.org/overriding-in-java/) compulsory OR either make subclass itself abstract.
5. Any class that contains one or more abstract methods must also be declared with abstract keyword.
6. There can be no object of an abstract class.That is, an abstract class can not be directly instantiated with the [*new operator*](https://www.geeksforgeeks.org/new-operator-java/).
7. An abstract class can have parametrized constructors and default constructor is always present in an abstract class.

**When to use abstract classes and abstract methods with an example**

There are situations in which we will want to define a superclass that declares the structure of a given abstraction without providing a complete implementation of every method. That is, sometimes we will want to create a superclass that only defines a generalization form that will be shared by all of its subclasses, leaving it to each subclass to fill in the details.

Consider a classic “shape” example, perhaps used in a computer-aided design system or game simulation. The base type is “shape” and each shape has a color, size and so on. From this, specific types of shapes are derived(inherited)-circle, square, triangle and so on – each of which may have additional characteristics and behaviors. For example, certain shapes can be flipped. Some behaviors may be different, such as when you want to calculate the area of a shape. The type hierarchy embodies both the similarities and differences between the shapes.



// Java program to illustrate the

// concept of Abstraction

abstract class Shape

{

String color;

// these are abstract methods

abstract double area();

public abstract String toString();

// abstract class can have constructor

public Shape(String color) {

System.out.println("Shape constructor called");

this.color = color;

}

// this is a concrete method

public String getColor() {

return color;

}

}

class Circle extends Shape

{

double radius;

public Circle(String color,double radius) {

// calling Shape constructor

super(color);

System.out.println("Circle constructor called");

this.radius = radius;

}

@Override

double area() {

return Math.PI \* Math.pow(radius, 2);

}

@Override

public String toString() {

return "Circle color is " + super.color +

"and area is : " + area();

}

}

class Rectangle extends Shape{

double length;

double width;

public Rectangle(String color,double length,double width) {

// calling Shape constructor

super(color);

System.out.println("Rectangle constructor called");

this.length = length;

this.width = width;

}

@Override

double area() {

return length\*width;

}

@Override

public String toString() {

return "Rectangle color is " + super.color +

"and area is : " + area();

}

}

public class Test

{

public static void main(String[] args)

{

Shape s1 = new Circle("Red", 2.2);

Shape s2 = new Rectangle("Yellow", 2, 4);

System.out.println(s1.toString());

System.out.println(s2.toString());

}

}

Output:

Shape constructor called

Circle constructor called

Shape constructor called

Rectangle constructor called

Circle color is Redand area is : 15.205308443374602

Rectangle color is Yellowand area is : 8.0

**Encapsulation vs Data Abstraction**

1. [Encapsulation](http://contribute.geeksforgeeks.org/encapsulation-in-java/) is data hiding(information hiding) while Abstraction is detail hiding(implementation hiding).
2. While encapsulation groups together data and methods that act upon the data, data abstraction deals with exposing the interface to the user and hiding the details of implementation.
3. **What is difference between getWindowHandle() and getWindowHandles() in selenium?**

***driver.getWindowHandle()***– It returns a handle of the current page (a unique identifier). It is used to **get the address of the current browser** where the control is and **return type is String**. Syntax is:

String handle = driver.getWindowHandle()

***driver.getWindowHandles()***– It returns a set of handles of the all the pages available. It is **used to get the address of all the open browser** and its **return type is Iterator<String>**. Syntax is:

<List> handles=driver.getWindowHandles();

**Return type: Page Address**

1. **What is difference between RemoteWebDriver and Webdriver ?**

**WebDriver:**

* WebDriver object represents the browser in Selenium. Using this object you can control the Web browser.
* It is an interface of the **org.openqa.selenium.\*** package. Upon instantiating the implementations of this class the browser will be launched.
* FirefoxDrive, ChromeDriver, InternetExplorerDriver, SafariDriver, OperaDriver, HtmlUnitDriver, **RemoteWebDriver** are few implementations of the WebDriver Interface.

**RemoteWebDriver:**

* This is the WebDriver object to control the desired browser in the node PC over the Grid using the Capability(settings) and Hostname or IP of the Selenium HUB running.
* Using the RemoteWebDriver object instantiation, you can control any browser in the node PC which is on Windows, Mac, Linux etc.. platforms. For eg:

**//Capability(settings) to execute the Node with window 8 and firefox ver32**

**DesiredCapabilities capability = new DesiredCapabilities.firefox();**

**capability.setVersion(“32.0”);**

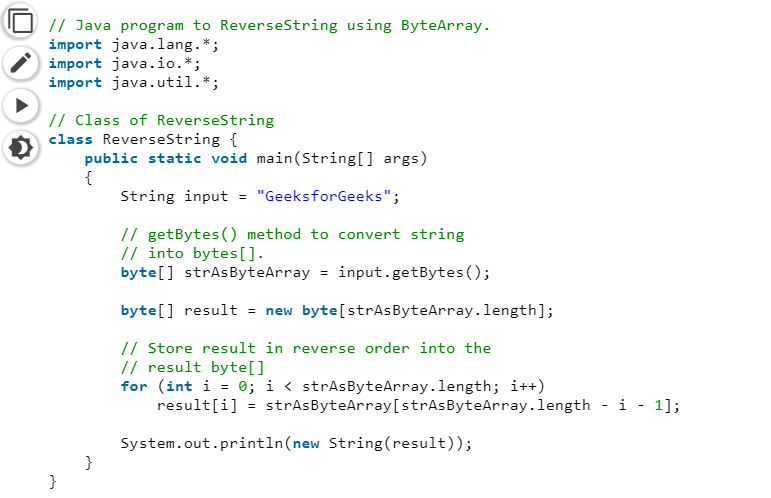
**capability.setPlatform(Platform.WIN8);**

**URL RegisteredUrl = new URL(“**[**http://10.10.10.10:4444/wd/hub**](http://10.10.10.10:4444/wd/hub)**”);**

**WebDriver driver = new WebDriver(RegisteredUrl, capability);**

**driver.get(“**[**https://www.edureka.co**](https://www.edureka.co/)**”);**

1. **Program to reverse a string in java?**



1. **What are the types of variable in java?**

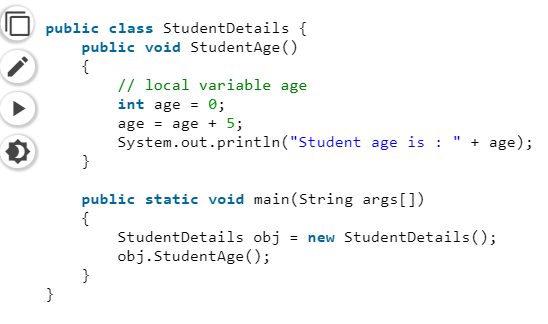
There are three types of variables in Java:

* Local Variables
* Instance Variables
* Static Variables

Let us now learn about each one of these variables in detail.

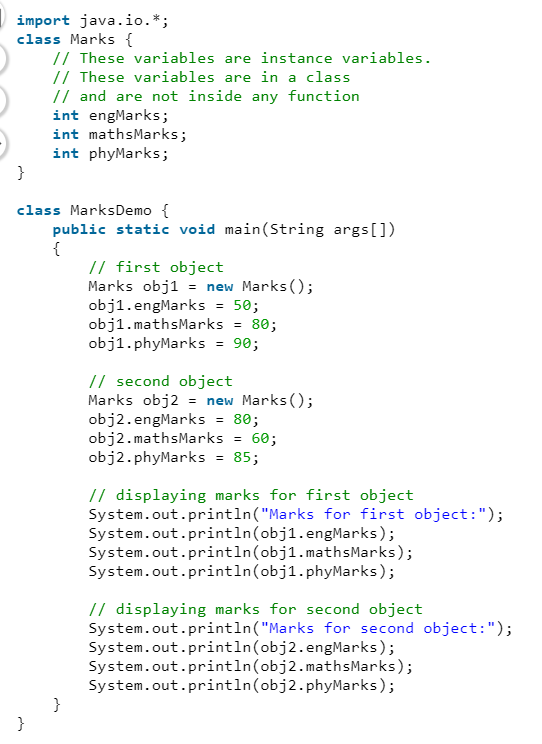
**Local Variables**: A variable defined within a block or method or constructor is called local variable.

* + These variable are created when the block in entered or the function is called and destroyed after exiting from the block or when the call returns from the function.
  + The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block.
  + Initilisation of Local Variable is Mandatory.



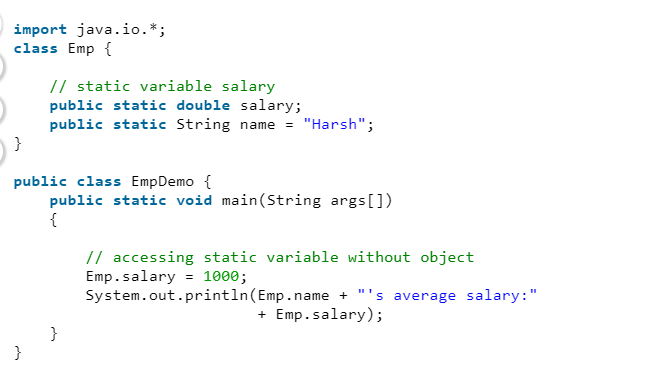
**Instance Variables**: Instance variables are non-static variables and are declared in a class outside any method, constructor or block.

* As instance variables are declared in a class, these variables are created when an object of the class is created and destroyed when the object is destroyed.
* Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier then the default access specifier will be used.
* Initilisation of Instance Variable is not Mandatory. Its default value is 0
* Instance Variable can be accessed only by creating objects.



**Static Variables**: Static variables are also known as Class variables.

* These variables are declared similarly as instance variables, the difference is that static variables are declared using the static keyword within a class outside any method constructor or block.
* Unlike instance variables, we can only have one copy of a static variable per class irrespective of how many objects we create.
* Static variables are created at the start of program execution and destroyed automatically when execution ends.
* Initilisation of Static Variable is not Mandatory. Its default value is 0
* If we access the static variable like Instance variable (through an object), the compiler will show the warning message and it won’t halt the program. The compiler will replace the object name to class name automatically.
* If we access the static variable without the class name, Compiler will automatically append the class name.

To access static variables, we need not create an object of that class, we can simply access the variable as

1. **What is the use of static in public static void main(Strng [] args)?**

**Java** program's main method has to be declared static because keyword static allows main to be called without creating an object of the class in which the main method is defined. If we omit static keyword before main **Java** program will successfully compile but it won't execute

1. Can we pass parameters to main java class, if not. Why?

You can write the public static void main() method with arguments other than String the program gets compiled.

Since the main method is the entry point of the Java program, whenever you execute one the JVM searches for the main method, which is public, static, with return type void, and a String array as an argument.

Public static void main(String args[]){

}

If anything is missing the JVM raises an error. Therefore, if you write a method with other data types (except String array) as arguments, at the time of execution, JVM does not consider this new method as the entry point of Java and generates an error.

## Example

In the following Java program, we are trying to use an integer array as arguments of the main method.

[Live Demo](http://tpcg.io/pChYr6)

public class MainExample {

   public static void main(int args[]) {

      System.out.println("Hello how are you");

 }

}

## Output

On executing, this program generates the following error −

Error: Main method not found in class MainMethodExample, please define the main

method as:

public static void main(String[] args)

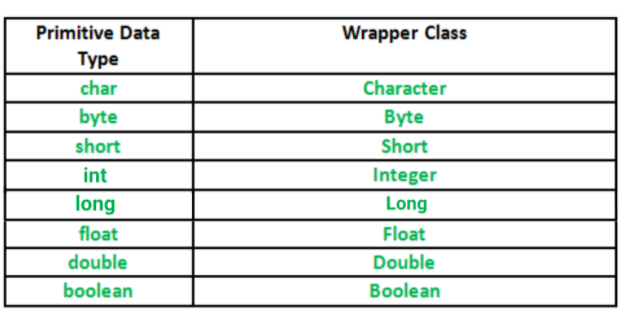
or a JavaFX application class must extend javafx.application.Application

1. **What are wrapper class in java? Why we use them?**

A Wrapper class is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

**Need of Wrapper Classes**

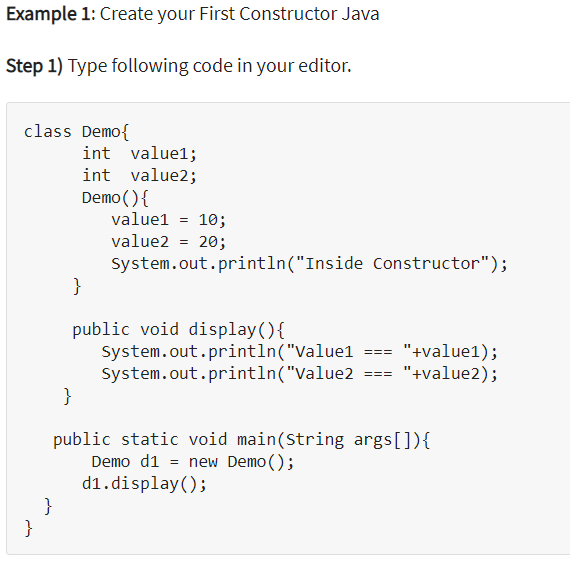
1. They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).
2. The classes in java.util package handles only objects and hence wrapper classes help in this case also.
3. Data structures in the Collection framework, such as [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/) and [Vector](https://www.geeksforgeeks.org/vector-vs-arraylist-java/), store only objects (reference types) and not primitive types.
4. An object is needed to support synchronization in multithreading.



1. **What are constructor in java?**

**CONSTRUCTOR** is a special method that is used to initialize a newly created object and is called just after the memory is allocated for the object. It can be used to initialize the objects to desired values or default values at the time of object creation. It is not mandatory for the coder to write a constructor for a class.

1. It has the **same name** as the class
2. It should not return a value not even ***void***



## Constructor Overloading

Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter list. The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

Examples of valid  constructors for class Account are

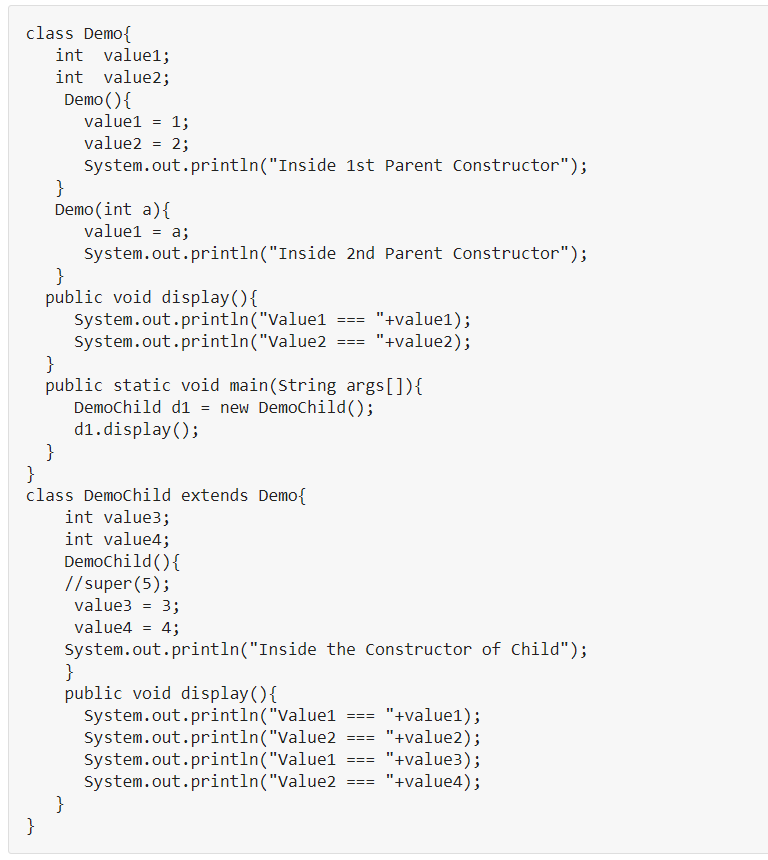
Account(int a);

Account (int a,int b);

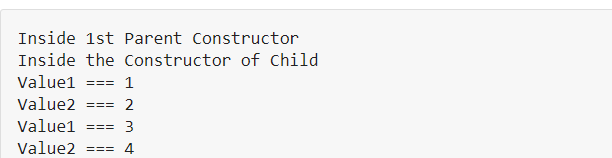
Account (String a,int b);

## Constructor Chaining

Consider a scenario where a base class is extended by a child. Whenever an object of the child class is created, the constructor of the parent class is invoked first. This is called **Constructor chaining.**



Run the Code. Owing to constructor chaining, when the object of child class DemoChild is created, constructor Demo() of the parent class is invoked first and later constructor DemoChild() of the child is created. Expected Output  =



1. **What are collections in java?**