Total-QA

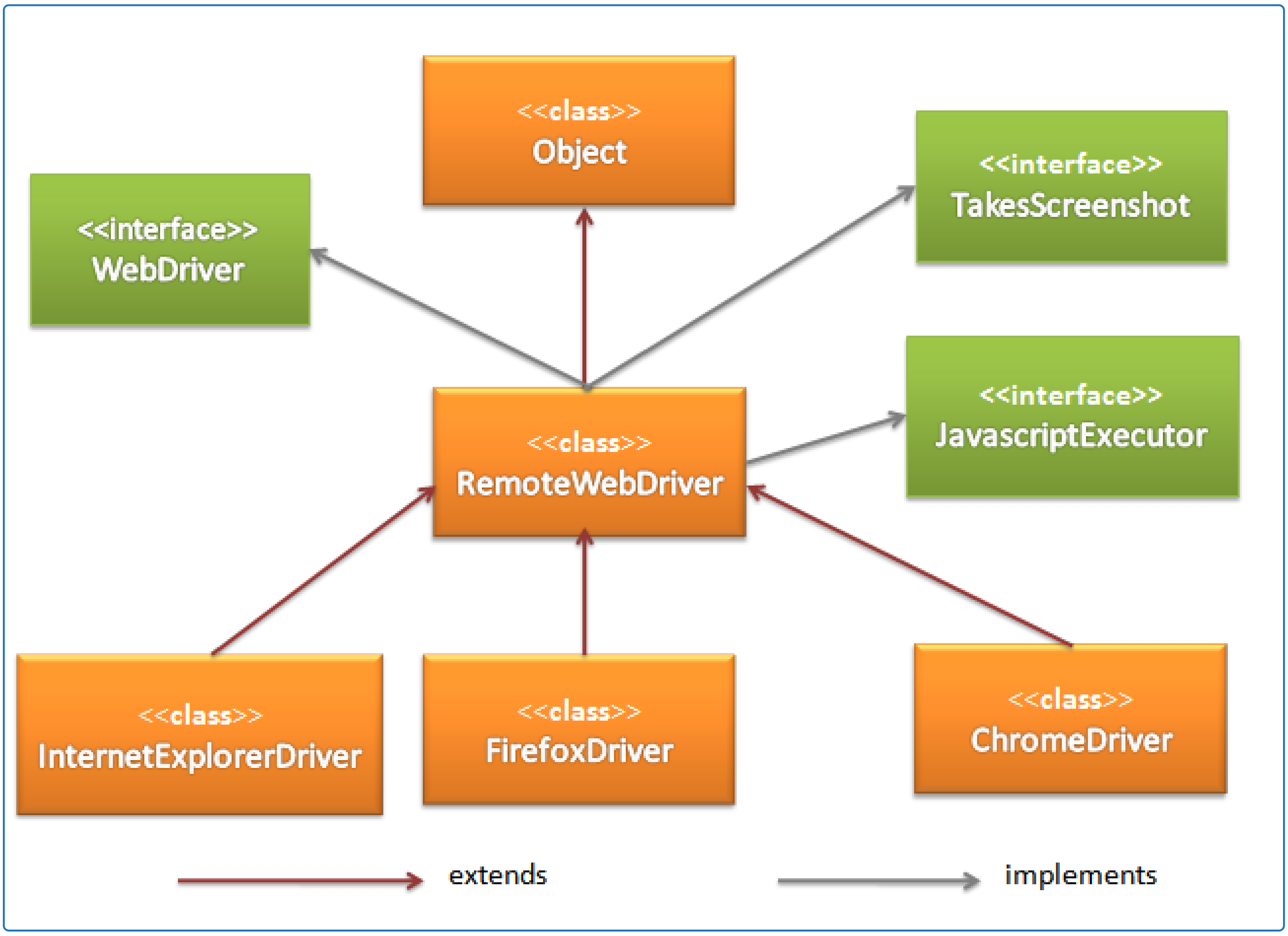
Future of Software Testing

Java OOPS

Refer to the below Links for Practise

- Collections in Java
- Complete List of Selenium Basics
- Top 50 Selenium Real Time Interview Questions
- Real Time Scenario’s
- Java Platform Standard Edition 8 Documentation
- Complete List of Java Basics
- Java Quiz
- Java OOPS Quiz
- Selenium Advanced Quiz
- Selenium Quiz
- Selenium WebDriver Resume

Inheritance,Abstract Class,Interfaces,Polymorphism



Inheritance:

Example->http://www.eclipse.org/eclipselink/documentation/2.5/concepts/img/javainhr.gif

```

1. The objective of inheritance is reusability.
2. The process of acquiring the methods and variables from SuperClass(RemoteWebDriver) to
   SubClass(FirefoxDriver) is called Inheritance.
   FirefoxDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
3. Inheritance is achieved between two classes using the keyword called 'extends'.
   So, from now onwards 'extends' means inheritance.

        Ex: Car is-a Vehicle
        Vehicle is superclass
        Car is subclass

                public class FirefoxDriver extends RemoteWebDriver
4.  Single Inheritance is between two classes is achieved using the extends keyword
5.  Multilevel Inheritance means the subclass(child) can access super class(Father)
```

```
    and super class can access super class of (Father) (GrandFather
) also.
6. Multiple Inheritance:
   In java we cannot extend more than one class at a time.
   Muliple classes cannot be inherited at the same time.
   So, multiple inheritance is not supported in java.

   ClassC extends ClassB,ClassA  //Not Valid in java

   Diamond Problem in java.

public class Object
{
    public int hashcode()
    {
        return 1;
    }
}

public class RWD extends Object
{
    public void get(String url)
    {

    }

}

public class FirefoxDriver extends RWD
{

}

public class IndeedTests
{

    main()
    {
        System.setPRoperty();
        FirefoxDriver driver = new FirefoxDriver();
        driver.get("");
    }
}
```

Abstract Class:

```

 Methods: In java they are two types of methods

    a. concrete methods:

           Methods which are having body are called as Concrete Met
hods

        //Concrete Method or complete Method
        public int add(int a,int b)
        {

            //Method body
        }
```

```
           If a method has curly braces mean, it  has body.. it is
implemented.
    b. abstract methods:

           Methods which are not having body are called as Abstract
Methods

      //Abstract Method or In-Complete Method
          public abstract int add(int a, int b);


Points to be Noted for Abstract classes:

1. The abstract method should be declared with keyword abstract.
        //InComplete Class
        public abstract class Car
                {

           //Concrete Method
           public void gears()
           {
           }
                       //abstract Method
           public abstract void engine();

        }
2. If a class contains one abstract method, then the class should be
 declared as abstract.
3. Inside a abstract class, we can develop both abstract and concrete
 method.
    Ex: abstract - engine,concrete-brakes()
4. If a class is declared as abstract its not mandatory to develop
abstract methods.***************
5. We cannot create an instance/object of abstract class, hence we
cannot
   access non-static members of the abstract class.-Ex: we cannot invoke
 wheeling
6. The static members of the abstract class can be referred using class
 name.
    Ex: By.name(""),By.className("")

        non-static at the object level.
        static at the class level.
7. A SubClass inheriting an abstract class should override all the
abstract
   methods of abstract class. Ex: engine

Summary:
static methods are invoked using className
non-static concrete methods using inheritance invoked within subclass
abstract methods using inheritance implemented in the subclass and
invoked within subclass..
```

Interface:

```

An interface is a reference type in Java, it is similar to class, it
 is a collection of abstract methods.
A class implements an interface, thereby inheriting the abstract methods
 of the interface.
```

```
Along with abstract methods an interface may also contain constants
and static methods.
It doesnt contain non-static methods

final int i=10;
     i=15; // Error final variables are constants..


   public class Shapes // circle,cylinder,square,rectangle
   {
    public abstract void area();
    public abstract void noofsides();
  }
 interface Shapes
 {
    void area();
    public abstract void noofsides();
 }

Points to be noted for interfaces:

1. An interface is a group of related methods with empty bodies.Ex:
Abstract Methods
2. All interface abstract methods are by default/optional abstract
and public.
    Ex: Default means hear it is optional
  Ex: void area()  or public abstract void area()
3. We cannot develop non static concrete methods inside interfaces.
4. So, in interfaces the methods are only public ,abstract,static.
No other keywords are allowed.
5. Interface variables has to be initialized at the time of declara
tion.
6. All the interface variables are by default public static final.
7. Interfaces are going to be implemented by subclass using a keywo
rd called 'implements'
8. A SubClass implmenting an interface should implement all the abs
tract methods
of the interface.
9. A class can implement multiple interfaces. Mulitiple inheritance
is achieved using interfaces



  ClassA implements interfaceA,interfaceB
  {




 }
--------------------------------------------------------------------
-----------------------------------


Implementing an interface by a class can be written as follows:

  Interface refvar = new ClassName();
  WebDriver driver = new FirefoxDriver();
    WebDriver driver = new ChromeDriver();
  WebDriver driver = new InternetExplorerDriver();


final variables are constants final int i =5;
          i=6; //not possible..
final class cannot be inherited.
   Example:
   public final class System{}
   public class Example extends System{} // not possible
```

```

final methods cannot be overridden.
```

Polymorphism:

Polymorphism

```
Polymorphism its an most imp oops concept.

Polymorphism means existence of Methods and objects in many forms .2 types of polymorphism in java.


Note: Compile Time checks for syntax. Run Time memory allocation happens for objects,variables.

1. Compile Time/Static Binding/Method Overloading:
-------------------------------------------------------

Existence of two or more methods with in a class which are having same name but having parameters differ in the following way:

                    a. Type of Arguments

                                         public void area(int a){
}
                                         public void area(float a
){}



                    b. Position of Arguments

                    public void area(int a,float b){}
                    public void area(float a,int b){}


                    c. No of Arguments

                    public void area(int a){}
                    public void area(int a,int b){}



Then we should call these methods are overloaded methods. Method invocation is mapped to method impl during compile Time.
Static Binding means the mapping is fixed it will not change during run time.
Method Overloading happens with in a class.

    for priting integer->    int i =5;
                System.out.println(i);
     for priting float->

                             float f =3.5f;
                             System.out.println(f);

Constructor Overloading:
----------------------------
Constructor is special type of method which is having same name as the classname, but it doesnt have any returntype.

                                    public class Student
```

```
            public class Student
            {

        Student()//Default Constructor
                {


                 }

        Student(int i )//Parameterized Constructor
                {


                }

        Student(float f )//Parameterized Constructor
                {


                }

            }

2. Run Time/Dynamic Binding/Method Overriding
------------------------
If a method which is having same name and same arguments availabl
e in superclass and subclass
are known are Overriden methods.
Method invocation mappedto Method implementation is changed duri
ng compile to run time. So, we call this
as run time polymorphism and Dynamic Binding.

Note:

     A SuperClass can store its subclass object.

     Animal a = new Bird();   Bird is-a Animal. 'is-a' an in
heritance relationship.


     public class Animal{}

     public class Bird extends Animal{}

    main()

    {

      Animal a = new Bird();

    }


      }




---------------------------------
--------------------
Overview of System.out.println()

public class System
```

```
{

    static PrintStream out =new  PrintStream();


}

public class PrintStream
{

    void println(int i)
    {

    }
}
```