

RE: Reg: Day 18 Notes

Senthil Kumar T <senthil.kumart@hcl.com>

Wed 10/25/2023 4:01 PM

To:Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>;Yerramreddy Sravani <yerramreddy.sravani@hcl.com>;Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>;Yeleti Vani <yeleti.vani@hcl.com>;Yaramothu Vasantha <yaramothu.vasantha@hcl.com>;Yallamma Arjun <yallamma.arjun@hcl.com>;Aashif Ameer <aashif.ameer@hcl.com>;Y Abhilash Reddy <yabhilash.reddy@hcl.com>;Aastha . <aastha.a@hcl.com>;Viyas Rameshkumar <viyas.rameshkumar@hcl.com>;Vishnu Vardhan S <vishnu.vardhans@hcl.com>;Vishal Kumar <vishalk@hcl.com>;Vipin Nonia <vipin.nonia@hcl.com>;Vinod . <vinod.v@hcl.com>;Vinithraj B <vinithraj.b@hcl.com>;Vinay Kumar Boora <vinaykumar_boora@hcl.com>;Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>;Abdul Suhana <abdul_suhana@hcl.com>;Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>;Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>

Classification: **Confidential**

Handson

1. Write hibernate appl to insert Person details like id,name,salary, address into database

ORM Framework

- Object Relational Mapping - used to persist the data into database

JDBC - Java Database Connectivity

EJB 2.x Entity bean - Failed in performance

EJB 3.X - JPA - Java Persistence API

JPA is a specification to persist the data into database and it uses some ORM concept

It requires some implementation using ORM tools like Hibernate, iBatis, Toplink, JDO etc

Object Relational Mapping(ORM) used to map POJO class(entity/persistent class) to the columns of database table

1. open source and lightweight
2. Database independent query
 - JPAQL(Java Persistent Query Language) - query the entity class
 - SQL - query the table directly
3. Automatic table creation
4. Fast performance - uses cache framework
5. Simplifies complex join
6. Hibernate will automatically generate the queries

1. Configure database info and hibernate properties

```
<!--Database info--!
<property name="connection.url">jdbc:mysql://localhost:3306/jpa</property>
<property name="connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="connection.username">root</property>
<property name="connection.password">root</property>

<!--Hibernate properties --!
<property name="hbm2ddl.auto">create/update/create-drop/validate</property> - will automatically create the table - 4 values
  1. create - each time it will create new table
  2. update - it will create table for first time, and next time it will update the table(ie) append the data
  3. create-drop - it drop the entire db and create a new table
  4. validate - it dosent do anything but it just validate the existing table
```

```
<property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
  - we can communicate with db using queries or using predefined methods
  - we wont write any sql queries, instead we use some predefined methods to processing in the database, so hibernate internally will generate sql queries related to that particular method based on particular database
<property name="show_sql">true</property>
  - used to display generated sql query in console in single line
<property name="format_sql">true</property>
  - used to format and display generated sql query in console
<property name="use_sql_comments">true</property>
  - used to define comments for the generated sql query in console
```

JPA - Java Persistence API

- It is a specification in Java used to persist data between java object and relational db
- collection of classes and interface to store data into database
- present in javax.persistence.*

Entity/persistent class(user defined class) - persistent object that is stored inside db - POJO class

1. EntityManagerFactory interface - used to create EntityManager
 - EntityManager createEntityManager()
2. EntityManager interface - core interface used to persist the data into db - acts as interface between appl and database
 - void persist(Object o) - insert the data into db if primary key does not exists, if primary key exists it perform update operation - generate insert query
 - Object find(Serializable s, Object key) - select the data from db - generate select query
 - void remove(Object o) - delete data from db - generate delete query
 - void close()
 - boolean contains(Object o)
 - Query createQuery(String query)
 - Query createNamedQuery(String query)
 - Query createNativeQuery(String query)
 - CriteriaBuilder getCriteriaBuilder()
 - EntityTransaction getTransaction() - create a transaction
3. EntityTransaction interface - used to maintain transaction - commit(), rollback()
4. Query interface - used to write database independent query

1. Create maven project with 2 dependency(hibernate, mysql db)

```
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>5.4.24.Final</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.19</version>
</dependency>
```

2. Configure db info in persistence.xml inside resources/META-INF

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="2.0">
  <persistence-unit name="student_pu" transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <class>com.pack.Student</class>
    <properties>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/prejoiners"/>
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.user" value="root"/>
      <property name="javax.persistence.jdbc.password" value="root"/>

      <property name="hibernate.hbm2ddl.auto" value="update"/>
      <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="true"/>
      <property name="hibernate.use_sql_comments" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

3. Create Entity class using JPA - simple pojo - use jpa concept

```
import javax.persistence.*;

@Entity - indicate it is entity class which will store in db
@Table(name="stud100") - Optional - used to map the entity class with table, if not given then it create table in name of entity class
public class Student {
  @Id - indicates it is a primary key
  @GeneratedValue(strategy=GenerationType.AUTO/IDENTITY/SEQUENCE/TABLE) - To indicate primary key value to be autogenerated, if we not
  provided @GeneratedValue we have to provide value for primary key

  @Column(name="studid",length=20,nullable="true/false",unique="true/false",insertable="true/false",updatable="true/false",scale="7",precision="2")
  private Integer id;
```

```

@Column(name="stuname")
private String name;

private Integer age;

@Temporal(TemporalType.DATE/TIME/TIMESTAMP) - used to map java date field to sql date field
private Date dob;

//private String gender;
@Enumerated(EnumType.ORDINAL(0,1)/STRING(MALE,FEMALE)) - used to map enum field to table
private Gender gender; //here gender is enum

@Lob - used to map large objects to table
@Column(name="photo",ColumnDefination="BLOB")
private byte[] myImage;

@Transient - used to ignore this field at time of persisting
private boolean status;
//getters and setters
}

@Column(name="sal",scale="7",precision="2")
private Double salary; //20000.50

enum Gender { MALE, FEMALE}

@Entity
@Table(name="stud100")
public class Student {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="stuid")
    private Integer id;
    @Column(name="stuname")
    private String name;
    private Integer age;
    @Temporal(TemporalType.DATE)
    private Date dob;
    @Enumerated(EnumType.ORDINAL)
    private Gender gender;
    @Transient
    private boolean status;

    //getters, setters,constructors
}

```

4. Configure entity class in persistence.xml

```
<class>com.pack.Student</class>
```

5. Create main class

1. Create EntityManagerFactory - get all db info from persistence.xml
2. Create EntityManager
3. Create EntityTransaction

@DynamicInsert - if we want to generate sql queries only for the column u r inserting
 @DynamicUpdate - if we want to generate sql queries only for the column u r updating

```

public class Main {
    public static void main(String[] args) {

        EntityManagerFactory emf=Persistence.createEntityManagerFactory("student_pu");
        EntityManager em=emf.createEntityManager();
        EntityTransaction et=em.getTransaction();
        et.begin();
        /*Student st1=new Student("Tim",23,LocalDate.parse("2000-10-23"),Gender.MALE);
        em.persist(st1);*/
    }
}

```

```

/*Student st=(Student)em.find(Student.class, 5);
st.setGender(Gender.FEMALE);
em.persist(st);*/

/*Student st=(Student)em.find(Student.class, 5);
em.remove(st);*/

/*Student st=new Student();
st.setName("Amy");
st.setAge(23);
st.setDob(LocalDate.parse("1990-02-23"));
st.setGender(Gender.FEMALE);
em.persist(st);*/

/*Student st=new Student();
st.setName("July");
st.setDob(LocalDate.parse("1993-02-23"));
st.setGender(Gender.FEMALE);
em.persist(st);*/

Student st=(Student)em.find(Student.class, 8);
st.setAge(28);
em.persist(st);
et.commit();
System.out.println("Success");
em.close();
emf.close();
}

}

```

From: Senthil Kumar T

Sent: Friday, October 20, 2023 3:33 PM

To: Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.an@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni < tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thaneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep_ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha < somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteeek <akasam.karteeek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silviu Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; 'Shuhail Rahman' <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti_sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala

<shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita .[akshita.a@hcl.com](#); Shashanki Sanjay Deshmane <shashanksanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyaK@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsrierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kumusa Kumari <seerakumusa.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer .<sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com>; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; 'Anagandula Sai Vardhan' <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; 'Rajesh Kethepalle' <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja_singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujavarapu <priyanka.jujavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r_priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare .<pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathiyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; 'Prakash Raj K' <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos .<anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; 'Ankur Prakash' <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; 'Nipun Samarakoon' <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita .<nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; 'Aparna M P' <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravindh L <aravindh.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakcharla Mounika <vakcharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrивastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhyा@hcl.com>; Mohd Fraz .<mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; 'Marcus Ivan Gonzalez Martinez' <marcusivan.gonzona@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullaipudi <mahesh.mullaipudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; 'Lohith N K' <lohitn.k@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; 'Likitha P' <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnapala <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati

Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumba_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <bhosalekshiti.somes@hcl.com>; 'Kshitij Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <athieeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondolu Srinidhi <kondolu.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasaki.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; 'Kannan Kumar T R' <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; kalpana.golla@hcl.com; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; JudsonSam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saiteja@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYAC <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashes@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvali Hemalatha <daluvali.hemalatha@hcl.com>; D Bommuraj <d.bommuraj@hcl.com>; Brinda Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; 'Calvin Jude Christy Paul' <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 18 Notes

Classification: **Confidential**

Spring Annotations

1. @Configuration
2. @Bean

```
public class Example3 {
    private String message;

    public Example3(String message) {
        super();
        this.message = message;
    }

    public Example3() {
        super();
    }
}
```

```

// TODO Auto-generated constructor stub
}

@Override
public String toString() {
    return "Example3 [message=" + message + "]";
}

}

@Configuration
public class ExampleConfiguration {

    @Bean
    @Scope("prototype")
    public Example1 example1() {
        return new Example1();
    }

    @Bean
    public Example2 example2() {
        return new Example2();
    }

    @Bean
    public Example3 example3() {
        return new Example3("Spring");
    }
    @Bean
    @Primary
    public Example3 example31() {
        return new Example3("Springboot");
    }
    @Bean
    public Example3 example32() {
        return new Example3("Struts");
    }
}

@Autowired
- perform autowiring based on by type
- used above only setter method or constructor or properties
- If we use autowire by annotation, in order to make ur spring to understand the annotation we have to configure <context:annotation-config/> in xml file

public class Address {
    private String city;
    private String state;
    private String pincode;
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
    public String getPincode() {
        return pincode;
    }
    public void setPincode(String pincode) {
        this.pincode = pincode;
    }
}

```

```

    }

public class Student {
    private Integer id;
    private String name;
    @Autowired
    private Address address; //referred another bean
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Address getAddress() {
        return address;
    }
    //Autowired
    public void setAddress(Address address) {
        this.address = address;
    }
}

```

{}

```

<context:annotation-config/>
<bean id="addr" class="com.pack.Address">
    <property name="city" value="Chennai"/>
    <property name="state" value="Tamilnadu"/>
    <property name="pincode" value="600011"/>
</bean>
<bean id="student" class="com.pack.Student">
    <property name="id" value="1000"/>
    <property name="name" value="John"/>
    <!-- <property name="address" ref="addr"/> -->
</bean>

```

```

ApplicationContext context=new ClassPathXmlApplicationContext("bean.xml");
Student st=(Student)context.getBean("student");
System.out.println(st.getName()+" "+st.getAddress().getCity());

```

@Qualifier

- If same bean is configured multiple times in xml file which bean has to be injected while using @Autowired is decided using @Qualifier

```

<context:annotation-config/>
<bean id="addr" class="com.pack.Address">
    <property name="city" value="Chennai"/>
    <property name="state" value="Tamilnadu"/>
    <property name="pincode" value="600011"/>
</bean>
<bean id="addr1" class="com.pack.Address">
    <property name="city" value="Pune"/>
    <property name="state" value="Maharastra"/>
    <property name="pincode" value="200011"/>
</bean>
<bean id="student" class="com.pack.Student">
    <property name="id" value="1000"/>
    <property name="name" value="John"/>
    <!-- <property name="address" ref="addr"/> -->
</bean>

```

```

public class Student {
    private Integer id;
    private String name;
    @Autowired
    @Qualifier("addr1")
    private Address address; //referred another bean
}

```

```

public Integer getId() {
    return id;
}
public void setId(Integer id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public Address getAddress() {
    return address;
}
//@Autowired
public void setAddress(Address address) {
    this.address = address;
}

}

}


```

@Required

- It indicates that the property should be compulsorily injected in xml file
- always used only above setter method

```

public class Student {
    private Integer id;
    private String name;
    @Autowired
    @Qualifier("addr1")
    private Address address; //referred another bean
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    @Required
    public void setName(String name) {
        this.name = name;
    }
    public Address getAddress() {
        return address;
    }
    //@Autowired
    public void setAddress(Address address) {
        this.address = address;
    }

}


```

BeanFactory is Lazy loading - it creates the bean only when we call getBean()

ApplicationContext is eager loading - preloads all the beans at time of startup itself, so we have to convert ApplicationContext to lazy loading

1. using lazy-init="true" in xml file
2. Using @Lazy annotation

```

public class Sample1 {

    public Sample1() {
        System.out.println("Sample1 bean is created");
    }

}

public class Sample2 {

}


```

```

private Sample1 sample1;

    public Sample1 getSample1() {
        return sample1;
    }

    public void setSample1(Sample1 sample1) {
        this.sample1 = sample1;
    }

    public Sample2() {
        System.out.println("Sample2 bean is created");
    }

}

<bean id="sample1" class="com.pack.Sample1" lazy-init="true"/>
<bean id="sample2" class="com.pack.Sample2" lazy-init="true">
    <property name="sample1" ref="sample1"/>
</bean>

public class Main3 {

    public static void main(String[] args) {
        //Resource res=new ClassPathResource("bean.xml");
        //BeanFactory factory=new XmlBeanFactory(res);
        //Sample2 s=(Sample2)factory.getBean("sample2");

        ApplicationContext ctx=new ClassPathXmlApplicationContext("bean.xml");
        Sample2 s=(Sample2)ctx.getBean("sample2");
    }
}

```

Lifecycle of spring bean

1. Using InitializingBean and DisposableBean interface

InitializingBean interface- public void afterPropertiesSet() - invoked whenever a bean is instantiated when getBean() is called
 DisposableBean interface - public void destroy() - invoked whenever ur bean is destroyed

ConfigurableApplicationContext interface
 - abstract class AbstractApplicationContext implements ConfigurableApplicationContext interface
 void registerShutdownHook() - destroy the bean
 or
 void close() - destroy the bean

```

public class Person implements InitializingBean,DisposableBean {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void destroy() throws Exception {
        System.out.println("Destroying person bean");
    }

    public void afterPropertiesSet() throws Exception {
        System.out.println("Initializing person bean "+name);
    }
}

<bean id="person" class="com.pack.Person">
    <property name="name" value="Ram"/>
</bean>

```

```
public class Main4 {

    public static void main(String[] args) {
        ConfigurableApplicationContext ctx=new ClassPathXmlApplicationContext("lifecycle.xml");
        Person p=(Person)ctx.getBean("person");
        System.out.println(p.getName());
        //ctx.registerShutdownHook();
        ctx.close();
    }
}
```

2. custom init and destroy method

- using init-method and destroy-method in xml file for particular bean class

```
public class Example {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void customInit() {
        System.out.println("Initializing example bean");
    }

    public void customDestroy() {
        System.out.println("Destroying example bean");
    }
}
```

```
<bean id="example" class="com.pack.Example" init-method="customInit" destroy-method="customDestroy">
    <property name="name" value="Sam"/>
</bean>
```

```
ConfigurableApplicationContext ctx=new ClassPathXmlApplicationContext("lifecycle.xml");
Example e=(Example)ctx.getBean("example");
System.out.println(e.getName());
ctx.registerShutdownHook();
```

3. default-init-method="customInit" and default-destroy-method="customDestroy" in beans tag, so that it is applicable for all beans in xml file

4. Using @PostConstruct and @PreDestroy annotation

```
<dependency>
    <groupId>javax.annotation</groupId>
    <artifactId>javax.annotation-api</artifactId>
    <version>1.3.2</version>
</dependency>
```

Spring stereotype annotations

- @Component - it allows spring to identify that class as custom bean so that it can be injected in other prg using @Autowired
- @Service - it is a bean class that contains business logic, so that it can be injected in other prg using @Autowired
- @Repository - it is a bean class that interacts with database
- @Controller - used to define a controller class that accept the request and gives the response

From: Senthil Kumar T

Sent: Monday, October 16, 2023 4:40 PM

To: Yerramsetti V Subrahmanyam Saisatyanarayana <yerramsetti.v.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh

Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalar.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Yamshi Krishna Bitla <yamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwal.arun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek.ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedhra <aherrutuja.rajedhra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at <hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeeli Chandran <siranjeeli.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh.a@hcl.com>; 'Shuhail Rahman' <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant.shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha.m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Sathesh Kumar S <satheesh.kumars@hcl.com>; Sasi Kumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <[sapkalutuja.pandu@hcl.com](mailto:sapkralutuja.pandu@hcl.com)>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer.s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshis@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit.kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepell <kallepell.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan.pandey@hcl.com>; Rijwana Parvez Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; 'Anagandula Sai Vardhan' <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; 'Rajesh Kethepalle' <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andhareartibaban@hcl.com>; Andri Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujavarapu <priyanka.jujavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; 'Prakash Raj K' <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khatti <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; 'Ankur Prakash' <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankaj Sharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar

<pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; 'Nipun Samarakoon' <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgupt@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; 'Aparna M P' <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravappalli Swarup <aravappalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravindh L <aravindh.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monisha@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhy@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; 'Marcus Ivan Gonzalez Martinez' <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumara.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; 'Lohith N K' <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; 'Likitha P' <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапala <ashini.rathnапala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; 'Kshitija Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha_s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpана Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; 'Kannan Kumar T R' <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; Kalpana.Golla <kalluru.sushma@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K P Argunam <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judsonam V <judsonam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jiJaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhima@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadarai S <jabadarai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <[">gudavally.saitje@hcl.com](mailto:gudavally.saitje@hcl.com)>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Varas Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh

Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathyv@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saitte@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; 'Calvin Jude Christy Paul' <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 17 Notes

Classification: **Confidential**

Scope

- life time of bean object - 5 types

1. singleton - default - how many bean object we are creating it will have same object
2. prototype - how many bean object we are creating it will have different reference
3. session
4. request
5. global-session

```
<bean id="emp" class="com.pack.Employee" scope="prototype">
    <constructor-arg index="0" type="java.lang.Integer" value="2000"/>
    <constructor-arg index="1" type="java.lang.String" value="Sam"/>
    <constructor-arg index="2" type="java.lang.Double" value="20000.0"/>
    <constructor-arg index="3" type="int" value="20"/>
</bean>
```

```
ApplicationContext context=new FileSystemXmlApplicationContext("hello.xml");
Employee e=(Employee)context.getBean("emp");
System.out.println(e);

Employee e1=(Employee)context.getBean("emp");
System.out.println(e1);

Employee e2=(Employee)context.getBean("emp");
System.out.println(e2);
```

Injection Collections

- apart from primitive datatype we can also inject collections like List, Set, Map, Properties

```
public class SampleCollection {
    private List addrList;
    private Set addrSet;
    private Map addrMap;
    private Properties addrProp;
    public List getAddrList() {
        return addrList;
    }
    public void setAddrList(List addrList) {
        this.addrList = addrList;
    }
    public Set getAddrSet() {
        return addrSet;
    }
    public void setAddrSet(Set addrSet) {
        this.addrSet = addrSet;
    }
    public Map getAddrMap() {
        return addrMap;
    }
    public void setAddrMap(Map addrMap) {
        this.addrMap = addrMap;
    }
    public Properties getAddrProp() {
        return addrProp;
    }
}
```

```

public void setAddrProp(Properties addrProp) {
    this.addrProp = addrProp;
}

}

<bean id="collection" class="com.pack.SampleCollection">
    <property name="addrList">
        <list>
            <value>Ram</value>
            <value>Sam</value>
            <value>Raj</value>
            <value>Ram</value>
        </list>
    </property>
    <property name="addrSet">
        <set>
            <value>Ram</value>
            <value>Sam</value>
            <value>Raj</value>
            <value>Ram</value>
        </set>
    </property>
    <property name="addrMap">
        <map>
            <entry key="1" value="Ram"/>
            <entry key="2" value="Sam"/>
            <entry key="3" value="Tam"/>
            <entry key="4" value="Raj"/>
        </map>
    </property>
    <property name="addrProp">
        <props>
            <prop key="one">Ram</prop>
            <prop key="two">Sam</prop>
            <prop key="three">Raj</prop>
        </props>
    </property>
</bean>
```

```

public class Main1 {
    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("bean.xml");
        SampleCollection s=(SampleCollection)ctx.getBean("collection");
        System.out.println(s.getAddrList());
        System.out.println(s.getAddrSet());
        System.out.println(s.getAddrMap());
        System.out.println(s.getAddrProp());
    }
}
```

ref attribute

- used to refer one bean into another bean
- ref="id of the bean"

```

public class Address {
    private String city;
    private String state;
    private String pincode;
    public String getCity() {
        return city;
    }
    public void setCity(String city) {
        this.city = city;
    }
    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
}
```

```

    }
    public String getPincode() {
        return pincode;
    }
    public void setPincode(String pincode) {
        this.pincode = pincode;
    }
}

public class Student {
    private Integer id;
    private String name;
    private Address address; //referred another bean
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Address getAddress() {
        return address;
    }
    public void setAddress(Address address) {
        this.address = address;
    }
}

<bean id="addr" class="com.pack.Address">
    <property name="city" value="Chennai"/>
    <property name="state" value="Tamilnadu"/>
    <property name="pincode" value="600011"/>
</bean>
<bean id="student" class="com.pack.Student">
    <property name="id" value="1000"/>
    <property name="name" value="John"/>
    <property name="address" ref="addr"/>
</bean>

public class Main1 {
    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("bean.xml");
        Student st=(Student)ctx.getBean("student");
        System.out.println(st.getId()+" "+st.getName()+" "+st.getAddress().getCity());
    }
}

```

wiring - referring one bean into another bean

Autowiring

- used only in case of referring one bean into another bean
- Instead of using ref attribute we can go for autowiring to inject one bean into another bean automatically

Types

1. autowire=none - default - where we have to do injection using ref attribute

```

public class Match {
    private String location;
    private String distance;
    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {

```

```

        this.location = location;
    }
    public String getDistance() {
        return distance;
    }
    public void setDistance(String distance) {
        this.distance = distance;
    }
}

public class Player {
    private Integer playerId;
    private String name;
    private Match match;
    public Integer getPlayerId() {
        return playerId;
    }
    public void setPlayerId(Integer playerId) {
        this.playerId = playerId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Match getMatch() {
        return match;
    }
    public void setMatch(Match match) {
        this.match = match;
    }
}

<bean id="match" class="com.pack.Match">
    <property name="location" value="Chennai"/>
    <property name="distance" value="1000"/>
</bean>
<bean id="player" class="com.pack.Player">
    <property name="playerId" value="100"/>
    <property name="name" value="Dhoni"/>
    <property name="match" ref="match"/>
</bean>

public class Main1 {
    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("bean.xml");
        Player p=(Player)ctx.getBean("player");
        System.out.println(p.getName()+" plays at "+p.getMatch().getLocation());
    }
}

```

2. autowire=byName - name of the property = id of the bean in xml file, then it will do autowire by name

```

<bean id="match" class="com.pack.Match">
    <property name="location" value="Chennai"/>
    <property name="distance" value="1000"/>
</bean>
<bean id="player" class="com.pack.Player" autowire="byName">
    <property name="playerId" value="100"/>
    <property name="name" value="Dhoni"/>
</bean>

```

3. autowire=byType - if datatype of the property should be somewhere configured in xml file, then it will do autowire by type

```

<bean id="match" class="com.pack.Match">
    <property name="location" value="Chennai"/>
    <property name="distance" value="1000"/>
</bean>

```

```
<bean id="player" class="com.pack.Player" autowire="byType">
    <property name="playerId" value="100"/>
    <property name="name" value="Dhoni"/>
</bean>
```

4. autowire=constructor - similar to byType or byName but it will do injection through constructor

```
public class Match {
    private String location;
    private String distance;
    public Match(String location, String distance) {
        super();
        this.location = location;
        this.distance = distance;
    }
    public Match() {
        super();
        // TODO Auto-generated constructor stub
    }
    @Override
    public String toString() {
        return "Match [location=" + location + ", distance=" + distance + "]";
    }
}

public class Player {
    private Integer playerId;
    private String name;
    private Match match1;
    public Player(Integer playerId, String name, Match match1) {
        super();
        this.playerId = playerId;
        this.name = name;
        this.match1 = match1;
    }
    public Player() {
        super();
        // TODO Auto-generated constructor stub
    }
    @Override
    public String toString() {
        return "Player [playerId=" + playerId + ", name=" + name + ", match1=" + match1 + "]";
    }
}

<bean id="match" class="com.pack.Match">
    <constructor-arg index="0" type="java.lang.String" value="Mumbai"/>
    <constructor-arg index="1" type="java.lang.String" value="1000"/>
</bean>
<bean id="player" class="com.pack.Player" autowire="constructor">
    <constructor-arg index="0" type="java.lang.Integer" value="1001"/>
    <constructor-arg index="1" type="java.lang.String" value="Kohli"/>
</bean>

public class Main1 {
    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("bean.xml");
        Player p=(Player)ctx.getBean("player");
        System.out.println(p);
    }
}

autowire-candidate=false
- If same bean is configured multiple times in XML file, which bean class has to be injected, it is defined using autowire-candidate property
- So which bean we don't want to use we have to define autowire-candidate=false
```

```
<bean id="match" class="com.pack.Match" autowire-candidate="false">
    <constructor-arg index="0" type="java.lang.String" value="Mumbai"/>
```

```

<constructor-arg index="1" type="java.lang.String" value="1000"/>
</bean>
<bean id="match1" class="com.pack.Match">
    <constructor-arg index="0" type="java.lang.String" value="Delhi"/>
    <constructor-arg index="1" type="java.lang.String" value="2000"/>
</bean>
<bean id="player" class="com.pack.Player" autowire="constructor">
    <constructor-arg index="0" type="java.lang.Integer" value="1001"/>
    <constructor-arg index="1" type="java.lang.String" value="Kohli"/>
</bean>

primary="true"
- If same bean is configured multiple times in xml file, which bean class has to be injected, it is defined using primary property
- So which bean we want to use we have to define primary="true"

```

```

<bean id="match" class="com.pack.Match" primary="true">
    <constructor-arg index="0" type="java.lang.String" value="Mumbai"/>
    <constructor-arg index="1" type="java.lang.String" value="1000"/>
</bean>
<bean id="match1" class="com.pack.Match">
    <constructor-arg index="0" type="java.lang.String" value="Delhi"/>
    <constructor-arg index="1" type="java.lang.String" value="2000"/>
</bean>
<bean id="player" class="com.pack.Player" autowire="constructor">
    <constructor-arg index="0" type="java.lang.Integer" value="1001"/>
    <constructor-arg index="1" type="java.lang.String" value="Kohli"/>
</bean>

```

Spring using Annotation

- used instead of configuring bean in xml file

@Configuration - class level annotation, something like xml file, where we configure all bean classes (ie) collection of bean classes

@Bean - used to configure single bean class

```

public class Example1 {
    private String message1;

    public String getMessage1() {
        return message1;
    }

    public void setMessage1(String message1) {
        this.message1 = message1;
    }
}

```

```

public class Example2 {
    private String message2;

    public String getMessage2() {
        return message2;
    }

    public void setMessage2(String message2) {
        this.message2 = message2;
    }
}

```

@Configuration
public class ExampleConfiguration {

```

    @Bean
    @Scope("prototype")
    public Example1 example1() {
        return new Example1();
    }
}

```

```

@Bean
public Example2 example2() {
    return new Example2();
}

}

public class Main2 {
    public static void main(String[] args) {
        ApplicationContext ctx=
            new AnnotationConfigApplicationContext(ExampleConfiguration.class);
        /*Example1 e1=(Example1)ctx.getBean(Example1.class);
        e1.setMessage1("Hello world");
        System.out.println(e1.getMessage1());

        Example2 e2=(Example2)ctx.getBean(Example2.class);
        e2.setMessage2("Welcome world");
        System.out.println(e2.getMessage2());*/

        Example1 e1=(Example1)ctx.getBean(Example1.class);
        System.out.println(e1);
        Example1 e2=(Example1)ctx.getBean(Example1.class);
        System.out.println(e2);
    }
}

```

From: Senthil Kumar T**Sent:** Friday, October 13, 2023 3:28 PM

To: Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Amer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vernavarapu Koteswara Rao <vernavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadi <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <vaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni < tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chowla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; 'sridhar.at@hcl.com' <sridhar.at@hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep_ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanth Pradhan <somanth.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti_sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav

<shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagya Chandra Patri <saubhagyanachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshishi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupeesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; 'ricardo.lopezporras@hcl.com' <ricardo.lopezporras@hcl.com>; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; 'Anagandula Sai Vardhan' <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetty Siva Krishna <ramisetty.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; 'Rajesh Kethepalle' <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; 'rahulvenkat.gone@hcl.com' <rahulvenkat.gone@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja_singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujavarapu <priyanka.jujavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r_priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathiyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; 'Prakash Raj K' <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khatti <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; 'Ankur Prakash' <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premisingh <padya.premsingh@hcl.com>; Pachimatl Laya <pachimatl.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; 'Nipun Samarakoon' <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoovrm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyu <arolla.anuhyu@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadaphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Athukuri <lolesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; 'Lohith N K' <lohith.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnapala <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar

<bhosalekshiti.somes@hcl.com>; 'Kshitij Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <attheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankar Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; 'Kannan Kumar T R' <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; 'kalpana.golla@hcl.com' <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; JudsonSam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitaja@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Dontu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati.bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashs@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <daluvai.hemalatha@hcl.com>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; 'Calvin Jude Christy Paul' <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 16 Notes

Classification: **Confidential**

Handson

1. Create User bean class with id, name, address, salary and inject the values using setter injection, constructor injection and print the output

J2EE - to develop web oriented appl - J2EE Components - Traditional approach

1. JSP - Java Server Pages - used to create dynamic web pages - UI Part
2. Servlets - java prg used to write business logic
3. EJB - Enterprise Java Bean - 3 beans
 1. Session bean - write business logic
 2. Entity bean - persist into db
 3. Message Driven bean - used to send sync and async messages

Problems with traditional approach

1. J2EE applications tend to contain excessive amounts of "plumbing" code -> There would always be a high proportion of code that doesn't do anything: JNDI lookup code, Transfer Objects, try/catch blocks to acquire and release JDBC resources. Writing and maintaining such plumbing code proves a major drain on resources that should be focused on the application's business domain.

2. J2EE applications are hard to unit test -> The J2EE APIs, especially, the EJB component model, does not take into account ease of unit testing. It is very difficult to test applications based on EJB and many other J2EE APIs outside an application server.

3. Certain J2EE technologies have failed in performance. EJB 2.x, for instance -> The main offender here is entity beans, which have proven little short of disastrous for productivity.

J2EE Frameworks

1. There are many frameworks which claim to resolve the issues mentioned earlier. For instance, Struts.

2. Struts is a web framework which works on the web tier and helps us achieve MVC and is doing pretty well in the market. However Spring takes over struts in that it is not just a web framework but an application framework.

3. Unlike single-tier frameworks such as Struts or Hibernate, Spring aims to help structure whole applications in a consistent, productive manner. It has modules that offer services for use throughout an application.

4. The essence of Spring is in providing enterprise services to Plain Old Java Objects (POJOs). This is valuable in a J2EE environment.

Spring framework

- open source framework, used to develop both standalone(Java) as well as Web(J2EE) appl. It is a framework so we have to download all jar files and put inside appl

Features

1. Reduce glue code/plumbing work

Spring Framework takes lot of load off the programmer by providing dependencies when required and by using AOP(Aspect Oriented Programming) - define common functionality in single place(bean) and access it anywhere

2. Externalize dependencies - Inversion of Control(IOC) - Dependency Injection

Dependencies are described in a separate file (xml) rather than mixing it with the business logic code itself. This gives a better control over the application.

Loosely coupled appl - inject the values at runtime defined in separate xml file

3. Manage dependencies at a single place

Dependencies can be managed better due to this.

4. Improve testability

Actual code can easily be replaced by a stub for testing purposes.

5. Foster good application design

Since the actual implementation sits behind the interfaces, it fosters good application design.

6. Flexibility

Spring offers integration points with several other frameworks. So, you do not have to write them yourself.

7 modules

1. Spring Core module

- support IOC concept
- Using BeanFactory interface

2. Spring Context module

- supports IOC concept + support internationalization, lifecycle events, integration with EJB, emailing, remoting, scheduling, integration with FreeMarker and Velocity framework
- Using ApplicationContext interface

3. Spring AOP(Aspect Oriented Programming)

- define common functionality in single place(bean) and access it anywhere

eg: security, transaction mgt, logging

4. Spring DAO(Data Access Object) module

- It provides some inbuilt interface to connect with database using Spring itself(JdbcTemplate class)

5. Spring ORM(Object Relational Mapping) module

- Used to integrate Spring with Hibernate framework

6. Spring Webflow module

- Used to integrate Spring with Struts framework

7. Spring MVC module

- used to develop MVC architecture using Spring itself

IOC - Inversion of Control

Dependency Injection

- objects are given their dependencies at runtime in separate xml file

3 types of DI

1. Setter injection - Spring

- dependencies are configured through beans setter methods for their properties using <property> tag

2. Constructor injection - Spring

- dependencies are configured through beans constructors for their properties using <constructor-arg> tag

3. Interface injection - Avalon

- dependencies are configured through interface for their properties

Spring IOC container - 2 types of IOC containers are there to perform DI

1. BeanFactory interface

- org.springframework.beans.factory.* package

- It is lazy loading

- used to create and destroy the bean

- Use XmlBeanFactory(Resource res) class to load the beans defined in xml file

- Object getBean(String name) - used to instantiate the bean and it performs DI

2. ApplicationContext interface

- org.springframework.context.* package

- It is eager loading

- used to create and destroy the bean

- 3 classes

1. ClassPathXmlApplicationContext(String xmlfilename) - loads the xml file from classpath(inside src folder)

2. FileSystemXmlApplicationContext(String xmlfilename) - loads the xml file from file system (ie) c:/ or d:/ or inside project

3. WebXmlApplicationContext(String xmlfilename) - loads the xml file from web.xml

```
public class Main {
    public static void main(String[] args) {
        //Resource res=new ClassPathResource("hello.xml");
        //Resource res=new FileSystemResource("hello.xml");
        //BeanFactory factory=new XmlBeanFactory(res); //load beans present in xml file
        //Student st=(Student)factory.getBean("stud");//create an object for bean which has id="stud" and perform DI
        //System.out.println(st.getName()+" "+st.getAge());

        //ApplicationContext context=new ClassPathXmlApplicationContext("hello.xml");
        ApplicationContext context=new FileSystemXmlApplicationContext("hello.xml");
        Student st=(Student)context.getBean("stud");
        System.out.println(st.getName()+" "+st.getAge());
    }
}
```

```
public class Student {
    private Integer id;
    private String name;
    private Integer age;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
}
```

```
public class Employee {
    private Integer id;
    private String name;
```

```

private Double salary;
private int age;
public Employee(Integer id, String name, Double salary, int age) {
    super();
    this.id = id;
    this.name = name;
    this.salary = salary;
    this.age = age;
}
public Employee() {
    super();
    // TODO Auto-generated constructor stub
}
@Override
public String toString() {
    return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", age=" + age + "]";
}

}

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<bean id="stud" class="com.pack.Student">
    <property name="id" value="1000"/>
    <property name="name" value="Ram"/>
    <property name="age" value="23"/>
</bean>

<!-- <bean id="emp" class="com.pack.Employee">
    <constructor-arg name="id" type="java.lang.Integer" value="2000"/>
    <constructor-arg name="name" type="java.lang.String" value="Sam"/>
    <constructor-arg name="salary" type="java.lang.Double" value="20000.0"/>
    <constructor-arg name="age" type="int" value="20"/>
</bean> -->

<bean id="emp" class="com.pack.Employee">
    <constructor-arg index="0" type="java.lang.Integer" value="2000"/>
    <constructor-arg index="1" type="java.lang.String" value="Sam"/>
    <constructor-arg index="2" type="java.lang.Double" value="20000.0"/>
    <constructor-arg index="3" type="int" value="20"/>
</bean>

</beans>

public class Main {
    public static void main(String[] args) {
        //Resource res=new ClassPathResource("hello.xml");
        //Resource res=new FileSystemResource("hello.xml");
        //BeanFactory factory=new XmlBeanFactory(res); //load beans present in xml file
        //Student st=(Student)factory.getBean("stud");//create an object for bean which has id="stud" and perform DI
        //System.out.println(st.getName()+" "+st.getAge());

        //ApplicationContext context=new ClassPathXmlApplicationContext("hello.xml");
        //ApplicationContext context=new FileSystemXmlApplicationContext("hello.xml");
        //Student st=(Student)context.getBean("stud");
        //System.out.println(st.getName()+" "+st.getAge());

        ApplicationContext context=new FileSystemXmlApplicationContext("hello.xml");
        Employee e=(Employee)context.getBean("emp");
        System.out.println(e);
    }
}

```

3 programs

1. Bean prg - simple pojo class contains getter and setter methods
2. xml file - used to perform dependency injection
3. main class

Apache Maven

- It is project development tool, instead of downloading the jar files manually, maven will download all the jar files on behalf of the user from internet
 - pom.xml(project object model) - we have to provide dependency
 - maven will contains 2 repo
1. Remote repository - for first time maven will download all jar files remotely present in internet and put in local repo
 2. Local repository - maven will internally create a folder .m2(local repo)

Installation

1. Download maven

<https://maven.apache.org/download.cgi>

Binary zip archive apache-maven-3.9.5-bin.zip

2. Extract apache maven

3. configure maven in env variable

In Search - Type env - Edit env variables for ur account

1. Under user variable - click New

Variable name: MAVEN_HOME

Variable value: C:\Softwares\apache-maven-3.6.3

2. Select Path - Click Edit - Click New - Paste C:\Softwares\apache-maven-3.6.3\bin

4. Open cmd prompt

>mvn -version

5. >mvn clean

- It will say build failure, used to create local repo .m2

goto c:/users/username/.m2

Spring project

1. Click File - New - Create Maven Project - Next

Check Create a simple project - Click Next

Group id: com.pack (package name)

ArtifactId: Spring1 (projectname)

Versioning: jar

Click Finish

2. Configure dependency in pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>4.2.4.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>4.2.4.RELEASE</version>
    </dependency>
</dependencies>
```

3. Right click project - click properties - Copy path from location - click apply and close

4. Open cmd prompt

C:\Users\senthil.kumart>cd C:\Spring\Spring1

C:\Spring\Spring1>mvn clean install

- It will download all jar files and keep inside local repo

5. Right click project - Maven - Update project - check Force update on snapshot - click ok

6. Create bean program inside src/main/java

```
public class Student {
    private Integer id;
    private String name;
    private Integer age;
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
}

public class Employee {
    private Integer id;
    private String name;
    private Double salary;
    private int age;
    public Employee(Integer id, String name, Double salary, int age) {
        super();
        this.id = id;
        this.name = name;
        this.salary = salary;
        this.age = age;
    }
    public Employee() {
        super();
        // TODO Auto-generated constructor stub
    }
    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", age=" + age + "]";
    }
}
```

7. Create xml file in any name inside src or inside project

- used to configure bean prg inside xml file to perform dependency injection
 - contain root elt called <beans> and each bean class is configured by <bean> tag
1. id - any name which acts as the object of bean class
 2. class - fully qualified path of bean prg

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context-3.0.xsd">
```

```

<bean id="stud" class="com.pack.Student">
    <property name="id" value="1000"/>
    <property name="name" value="Ram"/>
    <property name="age" value="23"/>
</bean>

<!-- <bean id="emp" class="com.pack.Employee">
    <constructor-arg name="id" type="java.lang.Integer" value="2000"/>
    <constructor-arg name="name" type="java.lang.String" value="Sam"/>
    <constructor-arg name="salary" type="java.lang.Double" value="20000.0"/>
    <constructor-arg name="age" type="int" value="20"/>
</bean> -->

<bean id="emp" class="com.pack.Employee">
    <constructor-arg index="0" type="java.lang.Integer" value="2000"/>
    <constructor-arg index="1" type="java.lang.String" value="Sam"/>
    <constructor-arg index="2" type="java.lang.Double" value="20000.0"/>
    <constructor-arg index="3" type="int" value="20"/>
</bean>

</beans>

```

8. Main class

Resource interface is used to locate the xml file - 2 classes

1. ClassPathResource class - if xml file present in src folder
2. FileSystemResource class - if xml file present in file system or project

```

public class Main {
    public static void main(String[] args) {
        //Resource res=new ClassPathResource("hello.xml");
        //Resource res=new FileSystemResource("hello.xml");
        //BeanFactory factory=new XmlBeanFactory(res); //load beans present in xml file
        //Student st=(Student)factory.getBean("stud");//create an object for bean which has id="stud" and perform DI
        //System.out.println(st.getName()+" "+st.getAge());

        //ApplicationContext context=new ClassPathXmlApplicationContext("hello.xml");
        //ApplicationContext context=new FileSystemXmlApplicationContext("hello.xml");
        //Student st=(Student)context.getBean("stud");
        //System.out.println(st.getName()+" "+st.getAge());

        ApplicationContext context=new FileSystemXmlApplicationContext("hello.xml");
        Employee e=(Employee)context.getBean("emp");
        System.out.println(e);
    }
}

```

From: Senthil Kumar T

Sent: Wednesday, October 11, 2023 3:17 PM

To: Yerramsetti V Subrahmanyam Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Amer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U

<swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajendra <aherrutuja.rajendra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeeli Chandran <siranjeeli.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvia Daniel Antohe Multescu <silviudaniel.antoh@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh .akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita .akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooy P <sayooy.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakuja <sanidhya.sakuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer .sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepell <kallepell.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; 'Rajesh Kethepalle' <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja_singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujjavarapu <priyanka.jujjavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r_priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare .pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathiyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; 'Prakash Raj K' <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos .anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornaya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; 'Ankur Prakash' <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hammant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatla Laya <pachimatla.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; 'Nipun Samarakoon' <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita .nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravindh L <aravindh.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>;

Arolla Anuhyा <arolla.anuhyaa@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonzja@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <lamthac.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапала <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <bhosalekshitji.somes@hcl.com>; 'Kshitija Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <krkitas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondouj Srinidhi <kondouj.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpана Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; 'Kannan Kumar T R' <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <katmatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; kalpana.golla@hcl.com; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitje@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Dontu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandeppalli Naveen <gandeppalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuv@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashes@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david.lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <daluvai.hemalatha@hcl.com>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammanna <chathuraruwan.gawar@hcl.com>; 'Calvin Jude Christy Paul' <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 16 Notes

Classification: **Confidential**

Handson

```
emp (eno(pk), ename, bdate, title, salary, dno)
proj (pno(pk), pname, budget, dno)
dept (dno(pk), dname, mrgeno)
workson (eno(fk), pno(fk), resp, hours)
```

Questions:

- 1) Write an SQL query that returns the project number and name for projects with a budget greater than \$100,000.
- 2) Write an SQL query that returns all works on records where hours worked is less than 10 and the responsibility is 'Manager'.
- 3) Write an SQL query that returns the employees (number and name only) who have a title of 'EE' or 'SA' and make more than \$35,000.
- 4) Write an SQL query that returns the employees (name only) in department 'D1' ordered by decreasing salary.
- 5) Write an SQL query that returns the departments (all fields) ordered by ascending department name.
- 6) Write an SQL query that returns the employee name, department name, and employee title.
- 7) Write an SQL query that returns the project name, hours worked, and project number for all works on records where hours > 10.
- 8) Write an SQL query that returns the project name, department name, and budget for all projects with a budget < \$50,000.
- 9) Write an SQL query that returns the employee numbers and salaries of all employees in the 'Consulting' department ordered by descending salary.
- 10) Write an SQL query that returns the employee name, project name, employee title, and hours for all works on records.

Subqueries

- select query inside an another select query
 - used to retrieve data based on unknown condition
 - Inner query will be executed first and based on output the outer query will be executed
 - 2 types of subqueries
1. Single row subquery - inner query returns a single row, then it is evaluated with outer query using =,!=,<,>,<=,>=
 2. Multi row subquery - inner query returns a multiple rows, then it is evaluated with outer query using any,all,in, not in, exists, not exists

```
mysql> create table emp(eid int primary key, firstname varchar(20), lastname varchar(20), city varchar(20));
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> insert into emp values(1,'Ram','Kumar','Chenna');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into emp values(2,'Sam','Kumar','Pune');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into emp values(3,'Saj','Kumar','Mumbai');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into emp values(4,'Raj','Kumar','Delhi');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into emp values(5,'John','Smith','Bangalore');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into emp values(6,'Jane','Baker','Hyderabad');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> create table dept(did int primary key, dname varchar(20), empid int references emp(eid), dsal float);
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> insert into dept values(1000,'HR',1,5000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept values(1001,'Sales',2,10000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept values(1002,'IT',3,5000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept values(1003,'Admin',4,10000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept values(1004,'HR',5,12000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept values(1005,'IT',6,15000);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from emp;
+----+-----+-----+
| eid | effirstname | elastname | ecity |
+----+-----+-----+
| 1 | Ram | Kumar | Chenna |
| 2 | Sam | Kumar | Pune |
| 3 | Saj | Kumar | Mumbai |
| 4 | Raj | Kumar | Delhi |
| 5 | John | Smith | Bangalore |
| 6 | Jane | Baker | Hyderabad |
+----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select * from dept;
+----+-----+-----+
| did | dname | empid | dsal |
+----+-----+-----+
| 1000 | HR | 1 | 5000 |
| 1001 | Sales | 2 | 10000 |
| 1002 | IT | 3 | 5000 |
| 1003 | Admin | 4 | 10000 |
| 1004 | HR | 5 | 12000 |
| 1005 | IT | 6 | 15000 |
+----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> select * from emp where eid=(select empid from dept where dsal=12000);
+----+-----+-----+
| eid | effirstname | elastname | ecity |
+----+-----+-----+
| 5 | John | Smith | Bangalore |
+----+-----+
1 row in set (0.01 sec)
```

```
mysql> select * from emp where eid in (select empid from dept where dsal>10000);
+----+-----+-----+
| eid | effirstname | elastname | ecity |
+----+-----+-----+
| 5 | John | Smith | Bangalore |
| 6 | Jane | Baker | Hyderabad |
+----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from emp where eid = (select empid from dept where dsal>10000);
ERROR 1242 (21000): Subquery returns more than 1 row
mysql> select * from emp where eid > any(select empid from dept where dsal<10000);
+----+-----+-----+
| eid | effirstname | elastname | ecity |
+----+-----+-----+
| 2 | Sam | Kumar | Pune |
| 3 | Saj | Kumar | Mumbai |
| 4 | Raj | Kumar | Delhi |
| 5 | John | Smith | Bangalore |
| 6 | Jane | Baker | Hyderabad |
+----+-----+
5 rows in set (0.01 sec)
```

```
mysql> select * from emp where eid > all(select empid from dept where dsal<10000);
+----+-----+-----+
| eid | effirstname | elastname | ecity |
+----+-----+-----+
| 4 | Raj | Kumar | Delhi |
| 5 | John | Smith | Bangalore |
| 6 | Jane | Baker | Hyderabad |
+----+-----+
3 rows in set (0.00 sec)
```

```
mysql> select * from emp where eid = all(select empid from dept where dsal<10000);
Empty set (0.00 sec)
```

```
mysql> select * from emp where exists(select * from dept where dsal=16000);
Empty set (0.00 sec)

mysql> select * from emp where not exists(select * from dept where dsal=16000);
+-----+-----+-----+
| eid | efirstname | elastname | ecity   |
+-----+-----+-----+
|  1 | Ram        | Kumar      | Chennai  |
|  2 | Sam        | Kumar      | Pune     |
|  3 | Saj        | Kumar      | Mumbai   |
|  4 | Raj        | Kumar      | Delhi    |
|  5 | John       | Smith      | Bangalore|
|  6 | Jane       | Baker      | Hyderabad|
+-----+-----+-----+
6 rows in set (0.00 sec)
```

Joins

- at the time of displaying we can join the table and get the data

Types

1. Equi join
 - we join the table based on some equality condition

```
mysql> create table dept1(deptid int primary key,dname varchar(20));
Query OK, 0 rows affected (0.13 sec)
```

```
mysql> insert into dept1 values(10,'Sales');
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into dept1 values(20,'Admin');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept1 values(30,'HR');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into dept1 values(40,'IT');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> create table emp1(empid int primary key,ename varchar(20),deptid int references dept1(deptid));
Query OK, 0 rows affected (0.11 sec)
```

```
mysql> insert into emp1 values(1,'Pat',10);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into emp1 values(2,'Sat',20);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into emp1 values(3,'Pack',10);
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into emp1(empid,ename) values(4,'Mack');
Query OK, 1 row affected (0.03 sec)
```

```
mysql> select * from dept1;
```

```
+-----+
| deptid | dname   |
+-----+
|  10 | Sales   |
|  20 | Admin   |
|  30 | HR      |
|  40 | IT      |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> select * from emp1;
```

```
+-----+
| empid | ename | deptid |
+-----+
|  1 | Pat  |  10 |
|  2 | Sat  |  20 |
+-----+
```

```
| 3 | Pack | 10 |
| 4 | Mack | NULL |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> select e.ename,d.dname from emp1 e, dept1 d where e.deptid=d.deptid;
+-----+
| ename | dname |
+-----+
| Pat | Sales |
| Sat | Admin |
| Pack | Sales |
+-----+
3 rows in set (0.00 sec)
```

2. Non equi join

- Join the table based on some non equality conditions like >,<,<=,>=,! =

3. Natural join

- create an implicit join clause based on common columns in a table

```
mysql> select * from emp1 natural join dept1;
+-----+
| deptid | empid | ename | dname |
+-----+
| 10 | 1 | Pat | Sales |
| 20 | 2 | Sat | Admin |
| 10 | 3 | Pack | Sales |
+-----+
3 rows in set (0.00 sec)
```

4. Inner join

- Similar to Equi join but we have to use "on" clause

```
mysql> select e.ename,d.dname from emp1 e inner join dept1 d on e.deptid=d.deptid;
+-----+
| ename | dname |
+-----+
| Pat | Sales |
| Sat | Admin |
| Pack | Sales |
+-----+
3 rows in set (0.00 sec)
```

5. Outer join

- used to display both matched and unmatched data

3 types

1. Left outer join

- all rows from left table + only matched rows right table

```
mysql> select e.ename,d.dname from emp1 e left join dept1 d on e.deptid=d.deptid;
+-----+
| ename | dname |
+-----+
| Pat | Sales |
| Sat | Admin |
| Pack | Sales |
| Mack | NULL |
+-----+
4 rows in set (0.00 sec)
```

2. Right outer join

- all rows from right table + only matched rows from left table

```
mysql> select e.ename,d.dname from emp1 e right join dept1 d on e.deptid=d.deptid;
+-----+
| ename | dname |
+-----+
| Pat | Sales |
| Sat | Admin |
+-----+
```

```
| Pack | Sales |
| NULL | HR   |
| NULL | IT   |
+-----+
5 rows in set (0.00 sec)
```

3. Full outer join

- used to display both matched and unmatched rows from both table
- In mysql, full outer join is not possible, we use union of left outer join and right outer join

```
mysql> select e.ename,d.dname from emp1 e left join dept1 d on e.deptid=d.deptid
-> union
-> select e.ename,d.dname from emp1 e right join dept1 d on e.deptid=d.deptid;
+-----+
| ename | dname |
+-----+
| Pat  | Sales |
| Sat  | Admin |
| Pack | Sales |
| Mack | NULL |
| NULL | HR  |
| NULL | IT  |
+-----+
6 rows in set (0.01 sec)
```

6. Cross join

- cartesian product of two table

```
>select * from emp1 cross join dept1;
or
>select * from emp1,dept1;
```

4. TCL - Transaction Control Language

- Transaction is a small unit of work that is performed against the database (ie) while DML operation
- 3 stmt

1. commit - used to commit the transaction performed in the database
2. rollback - undo all previous transaction before committing, only for DML

```
insert;
commit;
insert;
insert;
update;
rollback
```

3. savepoint - used to undo particular transaction

```
insert;
delete
create; - when we call DDL stmt inbetween, then autocommit will take place
insert;
rollback;
```

```
mysql> select * from emp1;
+-----+
| empid | ename | deptid |
+-----+
|  1 | Pat  |  10 |
|  2 | Sat  |  20 |
|  3 | Pack |  10 |
|  4 | Mack | NULL |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> insert into emp1 values(5,'Raj',20);
Query OK, 1 row affected (0.02 sec)
```

```
mysql> update emp1 set deptid=30 where empid=4;
```

Query OK, 1 row affected (0.02 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> commit;
```

Query OK, 0 rows affected (0.03 sec)

```
mysql> select * from emp1;
```

```
+-----+-----+
```

empid	ename	deptid
1	Pat	10
2	Sat	20
3	Pack	10
4	Mack	30
5	Raj	20

```
+-----+-----+
```

5 rows in set (0.00 sec)

```
mysql> insert into emp1 values(6,'Rajj',20);
```

Query OK, 1 row affected (0.01 sec)

```
mysql> create table emp2(id int,name varchar(20));
```

Query OK, 0 rows affected (0.28 sec)

```
mysql> START TRANSACTION;
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> SAVEPOINT initial;
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> insert into emp1 values(10,'Taj',30);
```

Query OK, 1 row affected (0.02 sec)

```
mysql> SAVEPOINT addition;
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> update emp1 set ename='Tajjjj' where empid=10;
```

Query OK, 1 row affected (0.02 sec)

Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> select * from emp1;
```

```
+-----+-----+
```

empid	ename	deptid
1	Pat	10
2	Sat	20
3	Pack	10
4	Mack	30
5	Raj	20
7	Saj	20
10	Tajjjj	30

```
+-----+-----+
```

7 rows in set (0.00 sec)

```
mysql> SAVEPOINT updation;
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> delete from emp1 where empid=10;
```

Query OK, 1 row affected (0.00 sec)

```
mysql> select * from emp1;
```

```
+-----+-----+
```

empid	ename	deptid
1	Pat	10
2	Sat	20
3	Pack	10
4	Mack	30
5	Raj	20
7	Saj	20

```
+-----+-----+
```

```
+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> SAVEPOINT deletion;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> rollback to updation;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from emp1;
```

```
+-----+-----+
| empid | ename | deptid |
+-----+-----+
| 1 | Pat | 10 |
| 2 | Sat | 20 |
| 3 | Pack | 10 |
| 4 | Mack | 30 |
| 5 | Raj | 20 |
| 7 | Saj | 20 |
| 10 | Tajjjj | 30 |
+-----+-----+
```

```
7 rows in set (0.00 sec)
```

```
mysql> rollback to addition;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from emp1;
```

```
+-----+-----+
| empid | ename | deptid |
+-----+-----+
| 1 | Pat | 10 |
| 2 | Sat | 20 |
| 3 | Pack | 10 |
| 4 | Mack | 30 |
| 5 | Raj | 20 |
| 7 | Saj | 20 |
| 10 | Taj | 30 |
+-----+-----+
```

```
7 rows in set (0.00 sec)
```

```
mysql> commit;
Query OK, 0 rows affected (0.03 sec)
```

5. DCL - Data Control Language

- GRANT - granting privilege to user
- REVOKE - removing privilege from user

```
>grant select,update on emp to system;
>revoke select,update on emp to system;
```

schema - anything that is stored into database (ie) table,views, procedure etc

Views

- It is a schema object
- It is called as virtual table because it doesn't have its own data, it will always contain data of another table, that table is called base table
- used to execute complex queries

2 types

1. Simple view
2. Complex view

Syntax: create [or replace] view viewname as select stmt;

```
mysql> select e.ename,d.dname from emp1 e,dept1 d where e.deptid=d.deptid;
```

```
+-----+
| ename | dname |
+-----+
| Pat | Sales |
| Sat | Admin |
| Pack | Sales |
| Mack | HR |
```

```
| Raj | Admin |
| Saj | Admin |
| Taj | HR  |
+-----+
7 rows in set (0.00 sec)
```

```
mysql> create view v1 as select e.ename,d.dname from emp1 e,dept1 d where e.deptid=d.deptid;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> select * from v1;
+-----+
| ename | dname |
+-----+
| Pat  | Sales |
| Sat  | Admin |
| Pack | Sales |
| Mack | HR  |
| Raj  | Admin |
| Saj  | Admin |
| Taj  | HR  |
+-----+
7 rows in set (0.02 sec)
```

```
mysql> show tables;
+-----+
| Tables_in_trainingbatch |
+-----+
| dept      |
| dept1     |
| emp       |
| emp1      |
| emp2      |
| v1        |
+-----+
6 rows in set (0.01 sec)
```

```
mysql> show full tables;
+-----+-----+
| Tables_in_trainingbatch | Table_type |
+-----+-----+
| dept      | BASE TABLE |
| dept1     | BASE TABLE |
| emp       | BASE TABLE |
| emp1      | BASE TABLE |
| emp2      | BASE TABLE |
| v1        | VIEW    |
+-----+-----+
6 rows in set (0.02 sec)
```

```
mysql> drop table emp1;
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> show full tables;
+-----+-----+
| Tables_in_trainingbatch | Table_type |
+-----+-----+
| dept      | BASE TABLE |
| dept1     | BASE TABLE |
| emp       | BASE TABLE |
| emp2      | BASE TABLE |
| v1        | VIEW    |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select * from v1;
ERROR 1356 (HY000): View 'trainingbatch.v1' references invalid table(s) or column(s) or function(s) or definer/invoker of view lack rights to use them
```

```
mysql> drop view v1;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> show full tables;
```

```
+-----+-----+
| Tables_in_trainingbatch | Table_type |
+-----+-----+
| dept      | BASE TABLE |
| dept1     | BASE TABLE |
| emp       | BASE TABLE |
| emp2     | BASE TABLE |
+-----+-----+
4 rows in set (0.00 sec)
```

Updatable views

- Certain views are updatable (ie) if we perform any DML operation on views it will affect the base table, but with following condition on select stmt
 1. select stmt should not contain distinct, set operator, aggregate function, group by clause, having clause
 2. left join, outer join
 3. subquery in select list

```
mysql> create view v2 as select * from dept1;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> insert into v2 values(60,'Finance');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from dept1;
```

```
+-----+-----+
| deptid | dname   |
+-----+-----+
|  10 | Sales   |
|  20 | Admin   |
|  30 | HR      |
|  40 | IT      |
|  50 | Accounts |
|  60 | Finance  |
+-----+-----+
6 rows in set (0.00 sec)
```

with check option in views

- Certain views are updatable, but at the time of updating the view, if we want to check some condition

```
mysql> create view v4 as select * from dept1 where deptid>50 with check option;
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> insert into v4 values(70,'Operation');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from v4;
+-----+-----+
| deptid | dname   |
+-----+-----+
|  60 | Finance |
|  70 | Operation |
+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> insert into v4 values(55,'Production');
Query OK, 1 row affected (0.03 sec)
```

```
mysql> insert into v4 values(45,'Development');
ERROR 1369 (HY000): CHECK OPTION failed 'trainingbatch.v4'
mysql> delete from v4 where deptid=55;
Query OK, 1 row affected (0.03 sec)
```

```
mysql> delete from v4 where deptid=10;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select * from dept1;
+-----+-----+
| deptid | dname   |
+-----+-----+
|  10 | Sales   |
|  20 | Admin   |
|  30 | HR      |
+-----+-----+
```

```
| 40 | IT      |
| 50 | Accounts |
| 60 | Finance |
| 70 | Operation |
+-----+
7 rows in set (0.00 sec)
```

Rename view
>rename table v3 to v4;

Dropping view
>drop view v3;

Alter the view
1. using create or replace
2. First drop the view and recreate the view once again

Index

- It is a schema object
- used for faster retrieval of data, if ur table contains huge number of data
- Index will be automatically create for primary key and unique key which is called as BTREE index

Syntax: create index indexname on tablename(columnname,....)

```
mysql> select * from dept1;
+-----+
| deptid | dname   |
+-----+
| 10    | Sales   |
| 20    | Admin   |
| 30    | HR      |
| 40    | IT      |
| 50    | Accounts |
| 60    | Finance |
| 70    | Operation |
+-----+
7 rows in set (0.00 sec)
```

```
mysql> create index idx1 on dept1(dname);
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> show indexes from dept1;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| dept1 | 0 | PRIMARY | 1 | deptid | A | 7 | NULL | NULL | BTREE | | | YES | NULL | |
| dept1 | 1 | idx1 | 1 | dname | A | 7 | NULL | NULL | YES | BTREE | | | YES | NULL | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.05 sec)
```

```
mysql> drop index idx1 on dept1;
Query OK, 0 rows affected (0.04 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> show indexes from dept1;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible | Expression |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| dept1 | 0 | PRIMARY | 1 | deptid | A | 7 | NULL | NULL | BTREE | | | YES | NULL | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

From: Senthil Kumar T

Sent: Monday, October 9, 2023 3:26 PM

To: Yerramsetti V Subrahmanyam Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>;

Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonja@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopal Reddy Koppula <venugopalareddy.koppu@hcl.com>; Venkatesh S <venkatesh@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadiadla <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.an@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <abhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvvu Swapna <duvvu.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjevi Chandran <siranjevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh.a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashank Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <shanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellulla <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kumara <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha.m@hcl.com>; Saubhagyam Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravananan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarasarwathi.rengu@hcl.com](mailto:saraswathi.rengu@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakhija <sanidhya.sakhija@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer.s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@scl.com>; Sakshi Dhone <sakshi.dhone@hcl.com>; Amit Kashyap <amit.kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amrutha Thakre <amrutha.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddì <ramu.doddi@hcl.com>; Ramisetty Siva Krishna <ramisetty.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhara Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavarapu <priyanka.juijavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; 'Prakash Raj K' <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khatti <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; 'Ankur Prakash' <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>;

Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanmant@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatla Laya <pachimatla.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; 'Nipun Samarakoon' <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravindh L <aravindh.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monisha@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhy@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonza@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <amamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапala <ashini.rathnапala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <bhosaliekshiti.somes@hcl.com>; 'Kshitija Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpiana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; 'Kannan Kumar T R' <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; Kalpana Golla <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amrita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K P Argunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijau.mallikarjun.lohar@hcl.com>; Jelio Hallies Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadarai S <jabadarai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitje@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri

Bhuvaneswari <enjamuri.bhava@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathyv@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashes@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david.lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvali Hemalatha <daluvali.hemalatha@hcl.com>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedula Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; 'Calvin Jude Christy Paul' <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 15 Notes

Classification: **Confidential**

Create the table WORLD CITY with the following data.

City	Country	Continent	Latitude	NorthSouth	Longitude	EastWest
Athens	Greece	Europe	37.59	N	23.44	E
Atlanta	United States	North America	33.45	N	84.23	W
Dallas	United States	North America	32.47	N	96.47	W
Nashville	United States	North America	36.09	N	86.46	W
Victoria	Canada	North America	48.25	N	123.21	W
Peterborough	Canada	North America	44.18	N	79.18	W
Vancouver	Canada	North America	49.18	N	123.04	W
Toledo	United States	North America	41.39	N	83.82	W
Warsaw	Poland	Europe	52.15	N	21.00	E
Lima	Peru	South America	12.03	S	77.03	W
Rio De Janeiro	Brazil	South America	22.43	S	43.13	W
Santiago	Chile	South America	33.27	S	70.40	W
Bogota	Colombia	South America	04.36	N	74.05	W
Buenos Aires	Argentina	South America	34.36	S	58.28	W
Quito	Ecuador	South America	00.13	S	78.30	W
Caracas	Venezuela	South America	10.30	N	66.56	W
Madras	India	Asia	28.36	N	77.12	E
Bombay	India	Asia	18.58	N	72.50	E
Manchester	England	Europe	51.30	N	0.0	null
Moscow	Russia	Europe	55.45	N	37.35	E
Paris	France	Europe	48.52	N	2.20	E
Shenyang	China	Asia	41.48	N	123.27	E
Cairo	Egypt	Africa	30.03	N	31.15	E
Tripoli	Lybia	Africa	32.54	N	13.11	E
Beijing	China	Asia	39.56	N	116.24	E
Rome	Italy	Europe	41.54	N	12.29	E
Tokyo	Japan	Asia	35.42	N	139.46	E
Sydney	Australia	Australia	33.52	S	151.13	E
Sparta	Greece	Europe	37.05	N	22.27	E
Madrid	Spain	Europe	40.24	N	3.41	W

Complete the following exercise:

- For all the different countries contained in the WORLD CITY table, display their names and the continent in which they are located. Make sure that no country name is duplicated.
- Write an SQL query to display the list of the city and country for all the cities that begin with letter R.
- Write an SQL query to display the list of the city and country for all the cities that end with letter A.
- Write an SQL query to display the list of the city and country for all the cities that begin with letter M and have exactly six letters in them.
- Write an SQL query to display the list of the city and country for all the cities that contain an A as the second letter.

Complete the following exercise:

- Create a following tables

CUSTOMER Table Schema:

COLUMN NAME	DESCRIPTION	DATATYPE	SIZE	CONSTRAINTS
CUSTID	Customer Id	Number		Primary Key
FNAME	First Name	Character	30	
LNAME	Last Name	Character	30	
ADDRESS	Customer Address	Character	50	
PHONENO	Phone	Number		Not Null
CITY	City	Character	20	
COUNTRY	Country	Character	20	
DATEFIRSTPURCHASED	Date of the first purchase by the customer	Date		
SUPPLIERID	Supplier Information	Number		Foreign Key

Data for CUSTOMER table:

CUSTID	FNAME	LNAME	ADDRESS	PHONENO	CITY	COUNTRY	DATEFIRSTPURCHASED	SUPPLIERID
1001	Das	Jeyaseelan	119, park Avenue, II street,	9841093428	Coimbatore	India	10-jan-2004	1
2001	Gopi	Govindraj	241, I floor, Kamaraj street, Madippakkam	9444124590	Chennai	India	25-mar-2005	4
1201	Dilip	Kishore	43, II Avenue, Anna Nagar	9997234534	Bangalore	India	20-aug-2004	2
1300	Aanand	Chowdhury	42/1 sector 1, II Street	9841054348	Bangalore	India	15-may-2005	2
1220	Chandra	Nagarajan	83, lal bagh	98410672356	Bangalore	India	12-feb-2006	4
1221	Abhishek	Kumar	13,kishori park,	94447623901	Chennai	India	15-may-2004	1
1320	Nikhil	Pandit	218, alwaanya street	94448923091	Salem	India	21-apr-2006	3
1222	Meenu	Monica	C11, church road	98410563421	Trichy	India	30-aug-2004	1
1225	Pavan	Kumar	128/A, North Mada Street	99934782103	maduari	India	18-aug-2004	4

SUPPLIER Table Schema:

COLUMN NAME	DESCRIPTION	DATATYPE	SIZE	CONSTRAINTS
SUPPLIERID	Supplier Id	Number		Primary Key
SNAME	Supplier Name	Character	30	
SCITY	Supplier City	Character	30	
SPHONE	Supplier Phone	Number		Not Null
EMAIL	Email id of Supplier	Character	50	Unique

Data for SUPPLIER table:

SUPPLIERID	SNAME	SCITY	SPHONE	EMAIL
1	Dilip	Chennai	8999900000	dilip@abc.co.in
2	Tarun	Madurai	8999911111	tarun@xyz.com
3	Naresh	Coimbatore	8999922222	g.naresh@xyzl.com
4	Ganesan	Trichy	8999933333	Ganesan_83@ijk.com

ORDERS Table Schema:

COLUMN NAME	DESCRIPTION	DATATYPE	SIZE	CONSTRAINTS
ORDERID	Order Number	Number		Primary Key
ORDERDATE	Date of Order	Date		
CUSTID	Customer Identity	Number		
QUANTITY	Quantity Ordered	Number		Check (Quantity >0)
ITEMID	Item Code	Number		Foreign Key

Data for ORDERS Table:

ORDERID	ORDERDATE	CUSTID	QUANTITY	ITEMID
1	12-jan-2004	1001	30	25
2	6-may-2005	1202	38	24
3	16-dec-2006	1220	10	22
4	21-may-2004	1233	12	21

ITEMS Table Schema:

COLUMN NAME	DESCRIPTION	DATATYPE	SIZE	CONSTRAINTS
ITEMID	Item Code	Number		Primary Key
ITEMNAME	Item Name	Character	35	Not Null
SUPPLIERID	Supplier Code	Number		Foreign Key
MINQTY	Minimum Qty that can be ordered	Number		Not Null
MAXQTY	Maximum Qty that can be ordered	Number		Not Null
Price	Price per unit	Number(5,2)		

Data for ITEMS Table:

ITEMID	ITEMNAME	SUPPLIERID	MINQTY	MAXQTY	Price
20	Pears Soap	4	7	20	30.00
21	V.V.D. Coconut oil 200 ml	2	8	15	79.00
22	Ponds powder 400g	3	6	25	106.00
23	Reynolds pen- blue	1	10	30	15.00
24	Reynolds pen- black	1	10	30	16.00
25	Mysore sandal soap	4	7	25	25.00
26	Fair & lovely cream- 50g	3	5	15	55.00
27	Rexono deo spary	2	5	20	100.00
28	Dove soap	4	7	15	85.00

b) Write SQL queries to

- Display all customers from Chennai.
- Display the details of all customers who purchased from the supplier 2.
- Display CUSTID, FNAME, LNAME of all customers whose purchase date is after January 2005.
- Display the details of all suppliers who are from location Coimbatore.
- Display the details of all suppliers whose name starts with G.
- Display the details of all customers, who do not have the alphabet e in their LNAME.
- Display the details of the entire customer whose first date of purchase is in 2006 and arrange it in descending order.
- Display the details of all the orders where the quantity is less than 35.
- Display the details of the items supplied by supplier 4.
- Display the details of all items where SUPPLIERID is 3 and the MINQTY is greater than 7 order by ITEMID.

SQL Statements

- DDL
- DML
- DQL - Data Query Language

1. select - used to select the data from table

> select * from customers; - select all rows and columns from table

> select custid, name from customers; - select all rows but only specific column

> select * from customers where custid>5; - select all columns but specific rows

>select custid,name from customers where custid>5; - select only specific column and specific rows

>select order.orderid, supplier.supname from order,supplier where order.orderid=supplier.orderid; - select from multiple table

>select o.orderid, s.supname from order o,supplier s where o.orderid=s.orderid; - create an alias for table to select data from tables multiple

>select distinct city from customers; - To avoid duplicates values

>select * from suppliers where city='Chennai' and type='PC manufacture';

>select * from suppliers where city='Chennai' or type='PC manufacture';

IS NULL, IS NOT NULL - to check null values

>select * from emp where dept is null;

>select * from emp where dept is not null;

Pattern Matching Operator - LIKE, NOT LIKE

- used to fetch data based on some pattern - applied only for varchar
- 2 wildcard characters
 - % - anything
 - _ - single value

>select * from suppliers where supname like 'R%'; - select all suppliers whose name should start with R

>select * from suppliers where supname like '%bob%';

>select * from suppliers where supname not like 'T%';

>select * from suppliers where supname like 'Sm_th';

>select * from orders where orderid like '1000_';

supname
Hello%
HelloHo
HelloBye
Hello%

>select * from suppliers where supname like 'Hello%'; - all 4 rows

If we want to include %,_ at time of pattern matching then we have to use escape sequence

>select * from suppliers where supname like 'Hello!%' ESCAPE '!'; - Return 2 rows

Gana
Girl
Giri
G%

>select * from suppliers where supname like 'G%';

>select * from suppliers where supname like 'G&%' ESCAPE '&';

Hello
Hello%%
Hi%

>select * from suppliers where supname like 'H%';

>select * from suppliers where supname like 'H%!%' ESCAPE '!';

>select * from suppliers where supname like 'H%\%';

IN, NOT IN - used of using multiple or operator

>select * from suppliers where supname='HCL' or supname='CTS' or supname='TCS';

>select * from suppliers where supname in('HCL','CTS','TCS');

>select * from orders where orderid not in(1000,2000,3000);

BETWEEN, NOT BETWEEN - Range operator - includes the ranges

```
>select * from suppliers where supid between 1000 and 2000;
```

```
>select * from suppliers where supid>=1000 and supid<=2000;
```

Set operator - UNION, UNION ALL, INTERSECT, MINUS

```
>select supid from supplier
```

```
union
```

```
select supid from order;
```

- select both supid from supplier and order table, if there is any duplication it will be considered only once

```
>select supid from supplier
```

```
union all
```

```
select supid from order;
```

- select both supid from supplier and order table, and print everything with duplication

```
>select supid from supplier
```

```
intersect
```

```
select supid from order;
```

- select only common supid from supplier and order table

```
>select supid from supplier
```

```
minus
```

```
select supid from order;
```

- remove supid from supplier that is present in order table

SQL Functions

1. Numeric function - apply these function to numbers

1. abs(n) - return absolute value

```
>select abs(9); 9
```

```
>select abs(-9); 9
```

2. ceil(n) - largest number >= n

```
>select ceil(9.5); 10
```

```
>select ceil(-9.5); -9
```

3. floor(n) - nearest number <= n

```
>select floor(9.5); 9
```

```
>select floor(-9.5); -10
```

4. mod(m,n) - mod(7,2) = 1

5. powe(m,n) - power(3,2) = 9

6. sqrt(n)

7. exp(n)

8. log(n)

9. sin(n)

10. cos(n)

11. tan(n)

12. round(n) - round to nearest decimal value

```
>select round(125.07); 125
```

```
>select round(125.78); 126
```

```
>select round(125.67,1); 125.7
```

```
>select round(125.11,1); 125.1
```

```
>select round(125.67,2); 125.67
```

```
>select round(125.678,2); 125.68
```

```
>select round(125.67,-1); 130    120 125 130
```

-1 represents 10's, if the value is greater than mid value it returns higher value otherwise it returns lower value

```
>select round(222.12,-1); 220    220 225 230
```

```
>select round(222.12,-2); 200    200 250 300
```

-2 represents 100's, if the value is greater than mid value it returns higher value otherwise it returns lower value

```
>select round(673.12,-2); 700    600 650 700
```

```
>select round(3567.12,-3); 4000   3000 3500 4000
```

-3 represents 1000's, if the value is greater than mid value it returns higher value otherwise it returns lower value

13. truncate(n)

```
>select truncate(125.13); error
```

```
>select truncate(125.13,1); 125.1
>select truncate(125.67,1); 125.6
>select truncate(125.678,2); 125.67
>select truncate(125.67,-1); 120
    -1 represents 10's, if the value is greater or lesser than mid value it always returns only lower value
>select truncate(435.12,-2); 400
```

2. Character function - apply only on varchar

1. ascii(n) - return ascii value of left most char

```
>select ascii('a'); 97
>select ascii('bad'); 98
```

2. char_length(n)
 character_length(n)
 length(n)
 - used to find length of string

```
>select char_length('hello'); 5
>select character_length('hello'); 5
>select length('hello'); 5
```

3. concat()/space operator - concatenate two or more string

```
>select concat('The answer is',24); The answer is 24
>select concat('The answer is', 10+10); The answer is 20
>select concat('The answer is', '10+10'); The answer is 10+10
>select concat('The answer is', null); null
>select 'The answer is' 24; The answer is 24
```

4. concat_ws() - concatenate two or more expr and adds a separator between each concatenated expr

```
>select concat_ws('+','a','b','c'); a+b+c
>select concat_ws('xyz','A','B','C'); AxyzBxyzC
>select concat_ws(null,'1','2'); null
```

5. field() - return the position of a value in a list of values

```
>select field('b','a','b','c','d','e','f'); 2
>select field(15,10,20,15,34); 3
>select field('c','a','b'); 0
>select field(null,'a','b'); null
```

6. find_in_set() - return the position of a value in a comma delimited string

```
>select find_in_set('b','a,b,c,d,e,f'); 2
>select find_in_set(2,'3,4,1,5,2'); 5
```

7. format() - used to format number with comma and rounding to nearest decimal places

```
>select format(123453.6789,2); 1,23,453.68
```

8. instr() - return location of substring in a string

```
>select instr('hello','h'); 1
>select instr('hello','e'); 2
>select instr('hello','i'); 0
```

9. lcase()
 lower()
 - convert string to lowercase

10. left() - extract leftmost char from a string

```
>select left('hello',1); h
>select left('hello',4); hell
>select left('hello',20); hello
```

11. right() - extract rightmost char from a string

```
>select right('hello',1); o
>select right('hello',4); ello
```

```
>select right('hello',20); hello
```

12. locate() - return the location of first occurrence of substring in a string

```
>select locate('H','Hello'); 1
>select locate('l','Hello'); 3
>select locate('l','Hello',3); 4 - from 3rd positin, first occurrence of l
>select locate('l','heelo',3); 4
>select locate('e','Technology internet'); 2
>select locate('e','technology internet',3); 15
>select locate('z','technology internet',3); 0
```

13. lpad(char1,n,char2) - char1 is left padded with char2 of length n

```
>select lpad('world',10,'hello'); helloworld
>select lpad('world',11,'hello'); hellohelloworld
>select lpad('world',12,'hello'); hellohelloworld
>select lpad('world',13,'hello'); hellohelelloworld
>select lpad('world',9,'hello'); hellworld
>select lpad('world',4,'hello'); worl
```

14. rpad(char1,n,char2) - char1 is right padded with char2 of length n

```
>select rpad('world',10,'hello'); worldhello
>select rpad('world',11,'hello'); worldhelloh
>select rpad('world',12,'hello'); worldhelloho
>select rpad('world',13,'hello'); worldhellohel
>select rpad('world',9,'hello'); worldhell
>select rpad('world',4,'hello'); worl
```

15. ltrim() - removes all spaces from left side

```
>select ltrim(' Hello '); Hello_
>select ltrim(' Hello world '); Hello world_
```

16. rtrim() - removes all spaces from right side

```
>select rtrim(' Hello '); _Hello
>select rtrim(' Hello world '); _Hello world
```

17. trim() - trims both leading and trailing space

```
>select trim(leading from ' hello '); hello_
>select trim(trailing from ' hello '); _hello
>select trim(both from ' hello '); hello
>select trim(leading '0' from '00012340'); 12340
>select trim(trailing '1' from '111hello111'); 111hello
```

18. ucase()

```
upper()
- convert string to uppercase
```

19. mid() - used to extract substring from a string

```
>select mid('technology',5,2); no
>select mid('technology',1,4); tech
>select mid('technology',-7,4); hnol
>select mid('technology',-3,3); ogy
```

20. substr() - used to extract substring from a string

```
>select substr('technology.com',5); nology.com
>select substr('technology.com',1,4); tech
>select substr('technology.com',-3,3); com
>select substr('technology.com' from 1 for 4); tech
```

21. substring() - used to extract substring from a string

```
>select substring('technology.com',5); nology.com
>select substring('technology.com',1,4); tech
```

```
>select substring('technology.com',-3,3); com
>select substring('technology.com' from 1 for 4); tech
```

22. substring_index() - return substring from string before number of occurrences of delimiter

```
>select substring_index('www.technology.com','.',1); www
>select substring_index('www.technology.com','.',2); www.technology
>select substring_index('www.technology.com','.',-1); com
```

23. strcmp() - compare 2 string

```
>select strcmp('hello','hello'); 0
>select strcmp('cat','hello'); -1
>select strcmp('hello','cat'); 1
```

24. space() - return a string with specified number of space

```
>select space(3); ''
```

25. reverse() - reverse the string

```
>select reverse('abcd'); dcba
```

26. replace()

```
>select replace('abc abc','a','B'); Bbc Bbc
```

3. Aggregate function/Group function

- used to group any value
- It will ignore the null value at a time of grouping
- sum(), avg(), max(), min(), count(), stddev(), median()
- For date datatype, we can apply only max(), min(), count()

Employee table

empid(int) name(varchar) gender(varchar) salary(double) dob(date) - consider we have totally 30 rows

What we provide in select list that only will be coming as column name in output

```
>select count(*) from employee;
```

```
count(*)
```

```
-----
```

```
30
```

```
>select count(gender) from employee;
```

```
count(gender)
```

```
-----
```

```
30
```

```
>select count(distinct gender) from employee;
```

```
count(distinct gender)
```

```
-----
```

```
2
```

Create an Alias for the column

```
>select count(*) as "Number of Employees" from employee;
```

Number of Employees

```
-----
```

```
30
```

```
>select count(*) "Number of Employees" from employee;
```

Number of Employees

```
-----
```

```
30
```

```
>select count(*) NumberofEmployees from employee;
```

NumberofEmployees

30

```
>select sum(salary) as "Total Salary" from employee;
>select max(dob) as DOB from employee;
>select min(dob) DOB from employee;
>select stddev(salary) from employee;
>select median(salary) from employee;
```

order by clause

- used for sorting purpose only at time of displaying
 - It should be always present in the last of select stmt
- select - from - where - group by - having - order by
- By default sorted in ascending order
 - column index represent the column from select list not from table
 - If we sort on two or more columns, then by default it will be sorted only on first column, in the first column if there is any repetition for those state column alone, the city will be sorted in desc order

```
>select supplier_city,supplier_state from supplier where supname='IBM' order by supplier_city;
```

```
>select supplier_city,supplier_state from supplier where supname='IBM' order by supplier_city asc;
```

```
>select supplier_city,supplier_state from supplier where supname='IBM' order by supplier_city desc;
```

Supplier table

sup_id supname supplier_city supplier_state address

```
>select supplier_city,supplier_state from supplier where supname='IBM' order by 2 desc; //column index represent the column from select list not from table
```

Supplier

supplier_city	supplier_state
Chennai	Tamilnadu
Allepey	Kerala
Coimbatore	Tamilnadu
Mumbai	Maharastra
Pune	Maharastra

```
>select supplier_city,supplier_state from supplier where supname='IBM' order by supplier_state asc, supplier_city desc;
```

Supplier

supplier_state	supplier_city
Kerala	Allepey
Maharastra	Pune
Maharastra	Mumbai
Tamilnadu	Coimbatore
Tamilnadu	Chennai

group by clause

- used to group the values in a table based on some aggregate function

Employee table

empid	ename	dept	salary
1	A	HR	4000
2	B	IT	6000
3	C	HR	6000
4	D	Admin	5000
5	E	IT	10000
6	F	Admin	6000
7	G	HR	5000

We want to calculate the total sum of salary

```
>select sum(salary) from employee;
```

We want to calculate the total sum of salary in each dept

```
>select dept,sum(salary) from employee group by dept;
```

dept sum(salary)

HR 15000

IT 16000
Admin 11000

having clause

- used to exclude the result from groupby clause

>select dept,sum(salary) from employee group by dept having sum(salary)>14000
order by dept asc;

```
dept  sum(salary)
HR    15000
IT    16000
```

From: Senthil Kumar T

Sent: Friday, October 6, 2023 3:17 PM

To: Yerramsetti V Subrahmanyam Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha ' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarrajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrry.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajednra <aherrutuja.rajednra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at <hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silviu Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha_m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakuja <sanidhya.sakuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyay <saktipada_u@hcl.com>; Sakshi Singh <sakshis@hcl.com>; Sakshi Dhone <sakshi.dhone@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana

Parveen Vakkapella <rjwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com>; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujjavapu <priyanka.jujjavapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankaj Sharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Prem singh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thinetla@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhy@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaf.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumara.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagridipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnapala <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwary@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshitji.somes@hcl.com](mailto:bhosalekshitji.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetty Pavan Kalyan <koppisetty.pavan@hcl.com>; Kondouji Srinidi <kondouji.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasaki.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakan.abhijit@hcl.com>; Kasipata Samarpiana Saraiyah <kasipatasmara.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumatr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.maland@hcl.com>; Kalpana Golla <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev

<kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jiju Mallikarjun Lohar <jijamallikar.lohar@hcl.com>; Jelio Halleye Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj . <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitje@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashs@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 14 Notes

Classification: **Confidential**

Complete the following assignment:

- a. Create the table PROGRAMMER with the given information using SQL CREATE TABLE command:

Attribute Name	Description/Data Type/Constraint
EmpNo	Employee's unique ID. Max. 5 characters should be numeric
ProjId	Project in which programmer participates. Max. 5 characters should be varchar
LastName	Surname of employee. Max. 30 characters. Required.
FirstName	Employee's first name. Max. 30 characters.
HireDate	Date on which employee was hired. Date data type.
Language	Programming language used by programmer. Max. 15 characters
TaskNo	Number of the task associated with the project. Numeric column, max. 2 digits
Privilege	Type of privilege given to programmer. Max. 25 characters.

- b) Insert the following data into the PROGRAMMER table:

EmpNo	LastName	FirstName	HireDate	ProjId	Language	TaskNo	Privilege
201	Gupta	Saurav	1/1/95	NPR	VB	52	Secret
390	Ghosh	Pinky	1/05/93	KCW	Java	11	Top Secret
789	Agarwal	Praveen	08/31/98	Rnc	VB	11	Secret
134	Chaudhury	Supriyo	07/15/95	TIPPS	C++	52	Secret
896	Jha	Ranjit	06/15/97	KCW	Java	10	Top Secret
345	John	Peter	11/15/99	TIPPS	Java	52	
563	Anderson	Andy	08/15/94	NIITS	C++	89	confidential

- c) Create a table WEATHER with following data:

City	State	High	Low
Calcutta	West Bengal	105	90
Trivandrum	Kerala	101	92
Mumbai	Maharashtra	88	69
Bangalore	Karnataka	77	60
New Delhi		80	72

d) Create a table BOOKS with the following data

BookId	Title	TopicId	Publisher Name	Placeof Publication	Price	PurchaseDate	ShelfNo
8293	DBMS	DB1	Prentice Hall	Mumbai	255	1/1/95	S11
5645	DBMS	DB1	Pearson Education	Mumbai	655	1/05/93	S12
6565	C	C1	TMH	Mumbai	840	08/31/98	S66
6567	C++	Cplus1	ABC Publishers	Delhi	300	07/15/95	S77
4576	JAVA	JAVA1	Guru Govind Publications	Delhi	500	06/15/97	S87
3433	OOPS	OOPS1	Dave Publishers	Pune	600	11/15/99	S56
4655	SAD	SAD1	Sajan Publications	Cochin	700	08/15/94	S76

Note:

- The primary keys and other keys could be assumed if not mentioned.
- When inserting values try using all the 3 different ways to insert data into the table

e) Write SQL queries to:

- i. Saurav Gupta is assigned a different project with id NITTTS and he would work with C++ now. Update this change in the PROGRAMMER table.
- ii. The books on DBMS are shifted to shelf with number S10. Please update this detail in BOOKS table.
- iii. Supriyo Chaudhury has resigned his job. Incorporate this change in the table PROGRAMMER.
- iv. A new column to state the nature of the climate with either of the value (rainy, cloudy, sunny, snow) is to be added in the WEATHER Table.

iv. Delete the table WEATHER from database.

Enum

- It is type of class that mainly stores constants or fixed set of values
- It implicitly abstract class, by default it is final so we can't inherit and it can implement by an interface

```

interface A {
    void add();
}

enum Day1 implements A {
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY;

    @Override
    public void add() {

    }
}

/*enum Day2 extends Day1{

}*/

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

public class Main {
    public static void main(String[] args) {
        Day d1=Day.TUESDAY;
        System.out.println(d1); //TUESDAY

        Day d2=d1;
        System.out.println(d2); //TUESDAY
        //Day d3=10; //error

        Day d3=Day.valueOf("MONDAY");
        System.out.println(d3); //MONDAY
    }
}

```

```
//Day d4=Day.valueOf("monday");
//System.out.println(d4); //IllegalArgumentException
System.out.println(d1==d2); //true
System.out.println(d1.equals(d2)); //true
System.out.println(Day.FRIDAY); //FRIDAY

for(Day d:Day.values()) {
    System.out.println(d.name()+" "+d.ordinal());
}
}
```

Rules

1. enum values are declared first, then only we can define anything

```
enum Day {
    //int a=10; //error
    SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY;
    int s=10; //correct
}
```

2. If we want to define anything after enum values then it should end with semicolon

3. A non final enum method can be overridden by any enum value

```
enum Day {
    SUNDAY {
        public void printTemp() {
            System.out.println("55");
        }
    },
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY;
    private String temperature;

    public void printTemp() {
        System.out.println("65");
    }
}
public class Main {
    public static void main(String[] args) {
        Day d1=Day.TUESDAY;
        d1.printTemp(); //65
        Day.SUNDAY.printTemp(); //55
    }
}
```

4. If we have abstract method then it should be implemented by all enum value

```
enum Day {
    SUNDAY {
        public void printTemp() {
            System.out.println("55");
        }
    },
    MONDAY {
        @Override
        public void printTemp() {
            System.out.println("50");
        }
    },
    TUESDAY {
        @Override
        public void printTemp() {
            System.out.println("45");
        }
    },
}
```

```

WEDNESDAY {
    @Override
    public void printTemp() {
        System.out.println("40");
    }
},
THURSDAY {
    @Override
    public void printTemp() {
        System.out.println("35");
    }
},
FRIDAY {
    @Override
    public void printTemp() {
        System.out.println("25");
    }
},
SATURDAY {
    @Override
    public void printTemp() {
        System.out.println("36");
    }
};
private String temperature;

    public abstract void printTemp();
}
public class Main {
    public static void main(String[] args) {
        Day d1=Day.TUESDAY;
        d1.printTemp(); //45
        Day.SUNDAY.printTemp(); //55
    }
}

```

5. Constructor is implicitly private, called once in the beginning to create enum value

```

enum Day {
    SUNDAY("High") {
        public void printTemp() {
            System.out.println("55");
        }
    },
    MONDAY("Low") {
        @Override
        public void printTemp() {
            System.out.println("50");
        }
    },
    TUESDAY("High") {
        @Override
        public void printTemp() {
            System.out.println("45");
        }
    },
    WEDNESDAY("Moderate") {
        @Override
        public void printTemp() {
            System.out.println("40");
        }
    },
    THURSDAY("Low") {
        @Override
        public void printTemp() {
            System.out.println("35");
        }
    },
    FRIDAY("High") {
        @Override
        public void printTemp() {

```

```

        System.out.println("25");
    }
},
SATURDAY("Low") {
    @Override
    public void printTemp() {
        System.out.println("36");
    }
};
String temperature;

/*private*/ Day(String temperature){
    this.temperature=temperature;
}

public abstract void printTemp();
}
public class Main {
    public static void main(String[] args) {
        Day d1=Day.TUESDAY;
        d1.printTemp(); //45
        Day.SUNDAY.printTemp(); //55

        for(Day d:Day.values()) {
            System.out.println(d+" "+d.temperature);
        }
    }
}

enum Marks {
    ENGLISH(67),
    MATHS(100),
    SCIENCE(99),
    SOCIAL(100),
    LANGUAGE(89);
    int mark;
    Marks(int mark){
        this.mark=mark;
    }
}
public class Main1 {

    public static void main(String[] args) {
        for(Marks m:Marks.values()) {
            System.out.println(m+" "+m.mark);
        }
    }
}

```

6. Enum can also used in switch case

```

public class Main {
    public static void main(String[] args) {
        Day1 d1=Day1.FRIDAY;
        switch(d1) {
        case SUNDAY:
            System.out.println("High");
            break;
        case MONDAY:
            System.out.println("Low");
            break;
        case TUESDAY:
            System.out.println("Moderate");
            break;
        default:
            System.out.println("No comments");
            break;
        }
    }
}

```

File processing system

- we store all data inside the files

Drawbacks

1. Data redundancy - duplication of data
 2. Data isolation - same data are scattered across multiple machines
 3. Only particular format of data can be stored inside the file
 4. No Security
 5. No backup and recovery of data
 6. Integrity constraints are buried under programming language
 7. Only single user can access the file

Database system

- set of logical related data - database
 - set of programs to access those data - DBMS/RDBMS

DBMS	RDBMS
1. No relationship between tables we have only primary key	1. We can relate two tables using foreign key concepts
2. Data is stored in flat file system	2. Data is stored inside the tables
3. Only single user can access the data at a time eg: MS Access, foxpro	3. Multiple user can access the data eg: MySQL, Oracle, SQL Server, PostgreSQL, DB2, Sybase

Data Models

- semantic representation of data

1. Hierarchical model
 2. Network model
 3. ER Model - Entity Relationship diagram
 4. Relational model - represent the data in the form of tables
Tables will contain rows(records/tuples) and columns(attributes/fields)
 5. Semistructured model - represent data in XML file

DBMS/RDBMS

- set a programs to access the data - 2 types
 1. SQL - Structured Query Language
 2. PLSQL - Procedural Language Structured Query Language

SQL

- It is a English like syntax to process the data in database

Types of keys in SQL

1. Super key - any combination of columns that uniquely fetch the value from database

Consider we have Employee table - empid,ename,address,salary,dept

ename,address,dept - super keys

empid,salary - candidate key (minimal of super key)

empid - primary key (anyone of candidate key is selected as primary key)

salary - alternate key

Integrity Constraints - 5 types

- 1. Entity integrity constraint
 - Entity means table - constraints that applied directly on the table
 - 2 types
 1. Primary key - no null value, no duplication
 2. Unique key - no duplication but it will allow null value
 - 2. Domain Integrity constraint
 - Domain is permitted value for the column
 - constraint that is applied to value for the column
 - 2 types
 1. NOT NULL
 2. CHECK

3. Referential Integrity Constraint

- We can refer one table with another table using foreign key concept
- One tables PK only will be acting as FK in another table, so first we have to create PK table then only we have to create foreign table
- Foreign key contains duplicate values and also contain null values
- FK column should contain only the values that are present in PK column
- whenever we have one to many, many to one, many to many relationship
- First we have to delete PK table then we can delete FK table

one person places many orders - one to many

Person table

pid(pk)	pname
1	A
2	B
3	C
4	D
5	E

Orders table

ordid(pk)	ordername	pid(FK)
100	Coffee	1
101	Tea	2
102	Chips	3
103	Juice	4
105	Biscuits	5
106	Cakes	1
107	Cola	

4. Default constraint

- By default if we didn't provide any value for the column, irrespective of the datatype always it will contain null value
- But we don't want to have null value, instead we want to provide some default value, then we can use default constraint

5. NULL constraint

Composite primary key

- If table contains more than one column as primary key where its combination should be unique

A(PK)	B(PK)	C
1	1	11
2	1	21
1	4	14
1	2	12

Note

1. By default if we didn't provide any value for the column, irrespective of the datatype always it will contain null value
2. SQL is a case sensitive - tablename, columnname is not case sensitivity, but the values stored inside the columns are case sensitive
'ram'!= 'Ram'

SQL Datatypes

1. Numeric datatype - store any numbers +ve or -ve

- numeric(p,q) p=total width, q=precision after decimal point
numeric(5,2) - 123.45
- numeric(p)
numeric(5) - 12345
- int
- integer
- tinyint -128 to 127
- smallint -32768 to 32767
- mediumint -8388608 to 8388607
- bigint -9223372036854775808 to 9223372036854775807
- decimal(p,q)/fixed(p,q)
- float
- double

2. Character datatype

- char(n) - fixed length of char
char(10)='ram'
- varchar(n) - variable length of char
varchar(10)='ram'

3. Date datatype

- date - only date(yyyy-MM-dd)
- datetime - both date and time (yyyy-MM-dd hh:mm:ss)
- timestamp - both date and time(yyyyMMddhhmmss)

4. LOB datatype - Large Object datatype

- BLOB(Binary Large Object) - used to store images, audio, video
- CLOB(Character Large Object) - used to store very long string

SQL Operators

1. +, -, *, /, <, >, <=, >=, !=

2. ALL

3. ANY

4. AND

5. OR

6. IN, NOT IN

7. LIKE, NOT LIKE - Pattern matching operator

8. BETWEEN, NOT BETWEEN - Range operator

9. IS NULL, IS NOT NULL

10. EXISTS, NOT EXISTS

11. UNION, UNION ALL, INTERSECT, MINUS - Set operator

mysql> show databases; - to view all databases

mysql> create database javabatch; - create a new db

Query OK, 1 row affected (0.03 sec)

mysql> use javabatch; - use the db

Database changed

mysql> show tables; - to view all tables in db

Empty set (0.02 sec)

mysql> drop database batch2; - delete the db

Query OK, 13 rows affected (0.24 sec)

SQL Statement

1. DDL statement - Data Definition Language - 4 stmt

- create, alter, drop, truncate

a. create - create a new table in db

mysql> create table customer(custid integer,name varchar(20),age int);

Query OK, 0 rows affected (0.07 sec)

mysql> desc customer; - to view structure of table

Field	Type	Null	Key	Default	Extra
custid	int	YES		NULL	
name	varchar(20)	YES		NULL	
age	int	YES		NULL	

3 rows in set (0.01 sec)

mysql> create table customer3(sno int unique,custid integer primary key AUTO_INCREMENT,name varchar(20),age int);

Query OK, 0 rows affected (0.07 sec)

mysql> create table customer4(sno int,custid integer,name varchar(20),age int,constraint cust_pk primary key(sno,custid));

Query OK, 0 rows affected (0.05 sec)

mysql> create table customer5(sno int unique,custid integer primary key AUTO_INCREMENT,name varchar(20) NOT NULL,age int default 25);

Query OK, 0 rows affected (0.06 sec)

mysql> create table department(deptid int primary key,dname varchar(20));

Query OK, 0 rows affected (0.06 sec)

mysql> create table employee(empid int primary key,ename varchar(20) NOT NULL, age int, did int references department(deptid));

Query OK, 0 rows affected (0.07 sec)

- Copy both data and structure from one table and create an another table

```
mysql> create table customer6 as select * from customer5;
```

Query OK, 0 rows affected (0.05 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> create table customer7 as select custid,name from customer5;
```

Query OK, 0 rows affected (0.05 sec)

Records: 0 Duplicates: 0 Warnings: 0

- Copy only structure from one table and create an another table

```
mysql> create table customer8 as select * from customer5 where 1=20;
```

Query OK, 0 rows affected (0.05 sec)

Records: 0 Duplicates: 0 Warnings: 0

b. alter

1. To rename table name

```
mysql> alter table customer8 rename to cust8;
```

Query OK, 0 rows affected (0.04 sec)

2. To rename column name

```
mysql> alter table cust8 rename column age to custage;
```

Query OK, 0 rows affected (0.01 sec)

Records: 0 Duplicates: 0 Warnings: 0

3. To add new column to exiting table

```
mysql> desc cust8;
```

Field	Type	Null	Key	Default	Extra
sno	int	YES		NULL	
custid	int	NO		0	
name	varchar(20)	NO		NULL	
custage	int	YES		25	

4 rows in set (0.00 sec)

```
mysql> alter table cust8 add address varchar(40);
```

Query OK, 0 rows affected (0.02 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> alter table cust8 add city varchar(40),add state varchar(40);
```

Query OK, 0 rows affected (0.03 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> desc cust8;
```

Field	Type	Null	Key	Default	Extra
sno	int	YES		NULL	
custid	int	NO		0	
name	varchar(20)	NO		NULL	
custage	int	YES		25	
address	varchar(40)	YES		NULL	
city	varchar(40)	YES		NULL	
state	varchar(40)	YES		NULL	

7 rows in set (0.00 sec)

```
mysql> alter table cust8 add email varchar(40) after custage;
```

Query OK, 0 rows affected (0.09 sec)

Records: 0 Duplicates: 0 Warnings: 0

```
mysql> desc cust8;
```

Field	Type	Null	Key	Default	Extra
-------	------	------	-----	---------	-------

```
| sno | int      | YES |   | NULL  |   |
| custid | int    | NO  |   | 0    |   |
| name  | varchar(20) | NO  |   | NULL  |   |
| custage | int    | YES |   | 25   |   |
| email  | varchar(40) | YES |   | NULL  |   |
| address | varchar(40) | YES |   | NULL  |   |
| city   | varchar(40) | YES |   | NULL  |   |
| state  | varchar(40) | YES |   | NULL  |   |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

4. Modify only datatype of existing column

```
mysql> alter table cust8 modify address varchar(50) NOT NULL;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> alter table cust8 modify city varchar(50) NOT NULL,modify state varchar(50) NOT NULL;
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
mysql> desc cust8;
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+
| sno  | int    | YES |   | NULL  |   |
| custid | int    | NO  |   | 0    |   |
| name  | varchar(20) | NO  |   | NULL  |   |
| custage | int    | YES |   | 25   |   |
| email  | varchar(40) | YES |   | NULL  |   |
| address | varchar(50) | NO  |   | NULL  |   |
| city   | varchar(50) | NO  |   | NULL  |   |
| state  | varchar(50) | NO  |   | NULL  |   |
+-----+-----+-----+-----+
```

5. To drop existing column in table

```
mysql> alter table cust8 drop column email;
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

6. To add constraint to the table after creating

```
mysql> alter table customer add constraint cu_pk primary key(custid);
Query OK, 0 rows affected (0.10 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

7. To enable/disable primary key or unique key

```
mysql> alter table customer disable keys;
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

```
mysql> alter table customer enable keys;
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

To enable/disable foreign key
SET FOREIGN_KEY_CHECKS=0;
SET FOREIGN_KEY_CHECKS=1;

c. drop

- drop the data as well as the table structure

```
mysql> drop table customer;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> drop table customer1,customer2;
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> drop table if exists cust8;
Query OK, 0 rows affected (0.03 sec)
```

d. truncate

- used to drop the data but the structure will be maintained but we cannot rollback

```
>truncate table customer4;
```

2. DML - Data Manipulation Language - 3 stmt

- insert, update, delete

a. insert - used to insert new row into the table

- insert all columns

```
mysql> insert into customer6 values(1,100,'Ram',24);
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into customer6 values(2,101,'Sam',25);
Query OK, 1 row affected (0.01 sec)
```

- insert only into particular columns

```
mysql> insert into customer6(sno,custid,name) values(3,103,'Raj');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select * from customer6;
```

sno	custid	name	age
1	100	Ram	24
2	101	Sam	25
3	103	Raj	25

3 rows in set (0.01 sec)

- insert data from another table, make sure both table should have same number of col and same column datatype

```
mysql> select * from customer5;
Empty set (0.00 sec)
```

```
mysql> insert into customer5 select * from customer6;
```

Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

```
mysql> select * from customer5;
```

sno	custid	name	age
1	100	Ram	24
2	101	Sam	25
3	103	Raj	25

3 rows in set (0.00 sec)

- with single insert query we want to insert multiple rows

```
mysql> insert into customer6(sno,custid,name,age) values
-> (4,104,'Tam',34),(5,105,'Tim',25),(6,106,'Jim',27);
Query OK, 3 rows affected (0.01 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

b. update - to update value of existing row

```
mysql> update customer6 set age=29 where custid=104;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
mysql> update customer6 set age=26,name='Jimmy' where custid=106;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

c. delete - used to delete the data from table but the structure will be maintained we can rollback

mysql> delete from customer6 where sno=6;
Query OK, 1 row affected (0.01 sec)

mysql> delete from customer6;
Query OK, 5 rows affected (0.01 sec)

From: Senthil Kumar T

Sent: Friday, September 29, 2023 3:28 PM

To: Yerramsetti V Subrahmanyamaisatyanarayana <yerramsetti.vsubra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha.' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisetty.vij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkatesh.s@hcl.com>; Venkata Suresh Alapa <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteshwara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi.hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwal.arun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek.ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneeru.syam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh.srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan.s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <swaraj.swarajmadhukar@hcl.com>; Duvvu Swapna <duvvu.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarrajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra.s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at <hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeesi Chandran <siranjeesi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silviu Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh.a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant.shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashank Sanjay Deshmane <shashanksanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzserra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settippalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha.m@hcl.com>; Saubhagyaa Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakuja <sanidhya.sakuja@hcl.com>; Sandeep Sirikonda <sandeep.sirkonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer.s@hcl.com>; Saktipada Upadhyay <saktipada.u@hcl.com>; Sakshi Singh <sakshis@hcl.com>; Sakshi Dhone <sakshi.dhone@hcl.com>; Amit Kashyap <amit.kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <rviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>;

Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavarapu <priyanka.juijavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pwankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vattcharla <vattcharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhy@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaf.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumara@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапала <ashini.rathnапала@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosalekshiti Somes <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritis@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondouji Srinidhi <kondouji.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasaki.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha.s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraih <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venka Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil.kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; Kalpana Golla <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kadde Amita Bhujangrao <kadde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagir <jyothipriya.ujjagir@hcl.com>; Judsonsam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini

<bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhima@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit.mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <prasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.sait@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaanilkumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <oopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvi Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brinda Vasudevan <brinda.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 13 Notes

Classification: **Confidential**

Handson

1.Display Date

Given a date in the form of string, write a program to convert the given string to date .

Include a class UserMainCode with a static method displayDate which accepts a string. In this method display the given string in date format yyyy-MM-dd. The return type is void.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of Date.

Refer sample output for formatting specifications.

Sample Input 1:

May 1, 2016

Sample Output 1:

2016-05-01

Sample Input 2:

March 21, 2016

Sample Output 2:

2016-03-21

2.Extract Date and time

Write a program to extract date and time from the input string which is in yyyy-MM-dd HH:mm:ss date format.

Include a class UserMainCode with a static method displayDateTime which accepts a string. In this method display date and time in the format as given in sample input and output . The return type is void.

Create a Class Main which would be used to accept string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output should be in date format

Refer sample output for formatting specifications.

Sample Input :

Enter String in this format(yyyy-MM-DD HH:mm:ss)

2016-07-14 09:00:02

Sample Output :

07/14/2016, 9:00:02

3.Day Of The Year

Given a date, write a program to display day of the year.

Include a class UserMainCode with a static method displayDay which accepts a date. In this method, display day in the format as given in sample input and output . The return type is void.?

?Input date Format is yyyy-MM-dd

Create a Class Main which would be used to accept date and call the static method present in UserMainCode.?

?

Input and Output Format:?

Input consists of a string.

Refer sample output for formatting specifications.

Sample Input :

2013-03-23

Sample Output :

Day of year: 82

4.Name Of the Day

Given a date in the date format, write a program to get the day of the corresponding date .

Include a class UserMainCode with a static method displayDay which accepts a date. In this method display the day of given date . The return type is void.

Create a Class Main which would be used to accept a date and call the static method present in UserMainCode.

?Input date Format is yyyy-MM-dd

Input and Output Format:

Input consists of a date.

Refer sample output for formatting specifications.

Sample Input 1:

2011-10-21

Sample Output 1:

Friday

Sample Input 2:

2011-07-11

Sample Output 2:

Monday

5.Difference between dates in month

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

Include a class UserMainCode with a static method getMonthDifference which accepts two date strings as input.

The return type of the output is an integer which returns the difference between two dates in months.

Create a class Main which would get the input and call the static method getMonthDifference present in the UserMainCode.

Input and Output Format:

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

2012-03-01

2012-04-16

Sample Output 1:

1

Sample Input 2:

2011-03-01

2012-04-16

Sample Output 2:

13

Before JDK1.8, Date class present in java.util.* package

1. Date class - print date and time
2. Calendar class - abstract class, used to extract useful info and date and time
3. GregorianCalendar class - concrete implementation of Calendar class, used to extract useful info and date and time
4. DateFormat class - abstract class, used for formating(convert date to string) and parsing(converting string to date), we can display date and time in different format - SHORT, LONG, MEDIUM, FULL - present in java.text.* pkg
5. SimpleDateFormat class - concrete implementation of DateFormat class, used for formating(convert date to string) and parsing(converting string to date), we can display date and time in different format - present in java.text.* pkg

```
public class Main {
```

```
    public static void main(String[] args) {
        Date d1=new Date(); //print current date and time
```

```
        System.out.println(d1); //Fri Sep 29 13:03:05 IST 2023
```

```
        Date d2=new Date(10000); //print date and time from Jan 1st 1970
```

```
        System.out.println(d2); //Thu Jan 01 05:30:10 IST 1970
```

```
        Calendar c=Calendar.getInstance(); //current date and time
```

```
        System.out.println(c.get(Calendar.YEAR));
```

```
        System.out.println(c.get(Calendar.MONTH));
```

```
        GregorianCalendar gc=new GregorianCalendar(); //current date and time
```

```
        System.out.println(gc.get(Calendar.YEAR));
```

```
        System.out.println(gc.get(Calendar.MONTH));
```

```
        GregorianCalendar gc1=new GregorianCalendar(2000,10,20,13,23,24);
```

```
        System.out.println(gc1.get(Calendar.HOUR));
```

```
        System.out.println(gc1.get(Calendar.MINUTE));
```

```
        DateFormat df=DateFormat.getInstance(); //default short format
```

```
        String s=df.format(d1);
```

```
        System.out.println(s);
```

```
        DateFormat df1=DateFormat.getDateInstance(DateFormat.LONG);
```

```
        String s1=df1.format(d1);
```

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

```
        DateFormat df2=DateFormat.getTimeInstance(DateFormat.MEDIUM);
```

```
        String s2=df2.format(d1);
```

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

```
        DateFormat df3=DateFormat.getDateTimeInstance(DateFormat.FULL,DateFormat.FULL); //default short format
```

```
        String s3=df3.format(d1);
```

```
        System.out.println(s3);
```

```
        String s4="10/03/2023";
```

```
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
```

```
        try {
```

```
            Date d3=sdf.parse(s4);
```

```
            System.out.println(d3); //Fri Sep 29 13:23:04 IST 2023
```

```
        SimpleDateFormat sdf1=new SimpleDateFormat("yyyy-MM-dd");
        System.out.println(sdf1.format(d3));
    }
    catch(ParseException e) {
        System.out.println(e);
    }
}
```

Date API

- Available from JDK1.8 onwards
 - present in `java.time.*` package
 - Date is immutable class where lower version of Date is not immuntable

class and interfaces

1. LocalDate class - print only date - yyyy-MM-dd
 2. TemporalAdjuster interface - used to get extra info from date and time
 3. LocalTime class - print only time
 4. LocalDateTime class - print both date and time
 5. Period class - used to find difference between 2 dates
 6. Duration class - used to find difference between 2 time
 7. DateTimeFormatter interface - used to display date and time in different formats

```
public class Main {  
    public static void main(String[] args) {  
        LocalDate l1=LocalDate.now(); //current date  
        System.out.println(l1); //2023-09-29  
  
        LocalDate l2=LocalDate.of(2000, 10,20); //create our own date  
        System.out.println(l2); //2000-10-20  
  
        LocalDate l3=LocalDate.parse("2001-04-23"); //convert string to date  
        System.out.println(l3); //2001-04-23  
  
        LocalDate l4=l3.plusMonths(2);  
        System.out.println(l4);  
        LocalDate l5=l4.plus(4, ChronoUnit.YEARS);  
        System.out.println(l5);  
  
        LocalDate l6=l5.minusDays(100);  
        System.out.println(l6);  
        LocalDate l7=l6.minus(5, ChronoUnit.WEEKS);  
        System.out.println(l7);  
  
        DayOfWeek d1=LocalDate.parse("2023-09-29").getDayOfWeek();  
        System.out.println(d1); //Friday  
        int m=LocalDate.parse("2023-09-29").getDayOfMonth();  
        System.out.println(m); //29  
        int y=LocalDate.parse("2023-09-29").getDayOfYear();  
        System.out.println(y); //272  
        int mon=LocalDate.parse("2023-09-29").getMonthValue();  
        System.out.println(mon); //9  
        System.out.println(l7.isLeapYear());  
  
        boolean b1=LocalDate.parse("2023-09-29").isAfter(LocalDate.parse("2023-09-30"));  
        System.out.println(b1); //false  
        boolean b2=LocalDate.parse("2023-09-29").isBefore(LocalDate.parse("2023-09-30"));  
        System.out.println(b2); //true  
        boolean b3=LocalDate.parse("2023-09-29").isEqual(LocalDate.parse("2023-09-30"));  
        System.out.println(b3); //false  
  
        LocalDate l8=LocalDate.now().with(TemporalAdjusters.firstDayOfMonth());  
        System.out.println(l8); //2023-09-01  
        LocalDate l9=LocalDate.now().with(TemporalAdjusters.lastDayOfMonth());  
        System.out.println(l9);  
        LocalDate l10=LocalDate.now().with(TemporalAdjusters.firstDayOfNextYear());  
        System.out.println(l10);  
        LocalDate l11=LocalDate.now().with(TemporalAdjusters.next(DayOfWeek.SATURDAY));  
        System.out.println(l11); //2023-09-30  
        LocalDate l12=LocalDate.now().with(TemporalAdjusters.previous(DayOfWeek.THURSDAY));  
        System.out.println(l12); //2023-09-28
```

```
LocalTime t1=LocalTime.now();
System.out.println(t1);
LocalTime t2=LocalTime.now(ZoneId.of("America/Chicago"));
System.out.println(t2);
LocalTime t3=LocalTime.of(9, 39);
System.out.println(t3);
LocalTime t4=LocalTime.parse("09:30");
System.out.println(t4);

LocalTime t5=t4.plusMinutes(23);
System.out.println(t5);
LocalTime t6=t5.plus(4, ChronoUnit.HOURS);
System.out.println(t6);
LocalTime t7=t6.minusNanos(20000);
System.out.println(t7);
LocalTime t8=t7.minus(35, ChronoUnit.SECONDS);
System.out.println(t8);
System.out.println(LocalTime.MAX);
System.out.println(LocalTime.MIN);
System.out.println(LocalTime.MIDNIGHT);

LocalDateTime ld1=LocalDateTime.now();
System.out.println(ld1); //2023-09-29T13:53:24
LocalDateTime ld2=LocalDateTime.of(LocalDate.now(), LocalTime.now());
System.out.println(ld2);
LocalDateTime ld3=LocalDateTime.parse("2023-09-29T13:53:24");
System.out.println(ld3); //2023-09-29T13:53:24
System.out.println(ld3.toLocalDate());
System.out.println(ld3.toLocalTime());
System.out.println(LocalDateTime.MAX);
System.out.println(LocalDateTime.MIN);

//convert JDK1.5 Date to JDK1.8 LocalDate
Date d11=new Date();
LocalDateTime ld4=LocalDateTime.ofInstant(d11.toInstant(), ZoneId.systemDefault());
System.out.println(ld4);

//convert Calendar to JDK1.8 LocalDate
Calendar cal=Calendar.getInstance();
LocalDateTime ld5=LocalDateTime.ofInstant(cal.toInstant(), ZoneId.systemDefault());
System.out.println(ld5);

LocalDate l13=LocalDate.now();
LocalDate l14=LocalDate.of(2023, 05, 20);
int diff1=Period.between(l13, l14).getDays();
System.out.println(diff1);

int diff2=Period.between(l13, l14).getMonths();
System.out.println(diff2);

long diff3=ChronoUnit.YEARS.between(l14, l2);
System.out.println(diff3);

LocalTime t9=LocalTime.now();
LocalTime t10=LocalTime.of(9, 30);
long diff4=Duration.between(t9,t10).getSeconds();
System.out.println(diff4);
long diff5=Duration.between(t9,t10).toHours();
System.out.println(diff5);
long diff6=Duration.between(t9,t10).toMinutes();
System.out.println(diff6);
long diff7=ChronoUnit.NANOS.between(t9,t10);
System.out.println(diff7);

LocalDateTime ld6=LocalDateTime.now();
String s1=ld6.format(DateTimeFormatter.ISO_DATE_TIME);
System.out.println(s1);
String s2=ld6.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
System.out.println(s2);
String s3=ld6.format(DateTimeFormatter.ofLocalizedDate(FormatStyle.SHORT));
```

```

        System.out.println(s3);
        String s4=Id6.format(DateTimeFormatter.ofLocalizedTime(FormatStyle.LONG));
        System.out.println(s4);
        String s5=Id6.format(DateTimeFormatter.ofLocalizedDateTime(FormatStyle.MEDIUM,FormatStyle.FULL));
        System.out.println(s5);
    }
}

```

Optional class

- Object will contain memory reference or null reference, so if we access anything using null reference we get NullPointerException
- To avoid NPE we have to write code to do the null check, so to avoid unpredictable NPE we can go for Optional class
- present in java.util.* package

Methods

1. static Optional empty() - return empty optional
2. static Optional of(T...t) - return an non empty optional, if value is null it return NPE
3. static Optional ofNullable(T...t) - return an non empty optional, if value is null it return empty optional
4. Object get() - return original value from Optional class
5. void ifPresent(Consumer) - if value is present it invokes related consumer
6. boolean isPresent() - return true if value is present otherwise false
7. Object orElse(Object) - return the value if present otherwise return other value
8. Object orElseGet(Supplier) - return the value if present otherwise it invoke another logic and return the result
9. Object orElseThrow(Supplier) - return the value if present otherwise it will throw an exception

```

public class Main {
    public static void main(String[] args) throws Throwable {
        Optional o1=Optional.empty();
        System.out.println(o1); //Optional.empty
        Optional o2=Optional.of("John");
        System.out.println(o2); //Optional[John]
        //Optional o3=Optional.of(null);
        //System.out.println(o3); //NPE

        Optional o4=Optional.ofNullable("Jim");
        System.out.println(o4); //Optional[Jim]
        Optional o5=Optional.ofNullable(null);
        System.out.println(o5); //Optional.empty

        Optional o6=Optional.of("John");
        System.out.println(o6); //Optional[John]
        System.out.println(o6.get()); //John
        o6.isPresent(System.out::println); //John
        System.out.println(o6.isPresent()); //true
        Optional o7=Optional.empty();
        System.out.println(o7.isPresent()); //false
        o7.isPresent(System.out::println); //no output
        System.out.println(o7.orElse("Johnny")); //Johnny
        System.out.println(o7.orElseGet(()->"Peter")); //Peter
        System.out.println(o7.orElseThrow(NullPointerException::new));
    }
}

```

3. Default and static methods in interface

- Till JDK1.7, interface will contain only abstract method and public static final variables
- From JDK1.8, apart from abstract method, interface can also contain default and static method
- Without affecting the implemented classes, if we want to add any new methods inside the interface then we can go for default methods
- If my functionality is noway related with object(ie) static method, instead of defining inside the class we can define inside the interface
- From JDK1.9, the interface will contain private and private static methods, if we have common functionality inside default method and to avoid duplicate code, we can define those common functionality inside private method and access inside default method whenever needed

```

interface A {
    default void show() {
        System.out.println("Inside A's show");
    }
}
interface B {
    default void show() {
        System.out.println("Inside B's show");
    }
}
class Sample implements A,B {

```

```

@Override
public void show() {
    //A.super.show();
    B.super.show();
}

}

public class Main {
    public static void main(String[] args) throws Throwable {
        Sample s=new Sample();
        s.show();
    }
}

```

Interface member access

1. Accessible from default and private methods within interface

constant variable = yes
 abstract method=yes
 another default method =yes
 private method=yes
 static method=yes
 private static method=yes

2. Accessible from static methods within interface

constant variable = yes
 abstract method=NO
 another default method =NO
 private method=NO
 static method=yes
 private static method=yes

3. Accessible from instance methods by implementing interface

constant variable = yes
 abstract method=yes
 another default method =yes
 private method=no
 static method=yes
 private static method=no

4. Accessible outside interface without implementing interface

constant variable = yes
 abstract method=no
 another default method =no
 private method=no
 static method=yes
 private static method=no

```

interface MyInterface {
    //Till JDK1.7
    /*public static final*/ int CONSTANT=0;
    /*public abstract*/ int abstractMethod();

    //From JDK1.8
    /*public*/ default int defaultMethod() {
        abstractMethod();
        privateMethod();
        staticMethod();
        privateStaticMethod();
        return CONSTANT;
    }
    /*public*/ static int staticMethod() {
        privateStaticMethod();
        return CONSTANT;
    }

    //From JDK1.9
    private int privateMethod() {
        abstractMethod();
        defaultMethod();
        staticMethod();
    }
}

```

```

        privateStaticMethod();
        return CONSTANT;
    }
    private static int privateStaticMethod() {

    }
}
class Example implements MyInterface {

    @Override
    public int abstractMethod() {
        defaultMethod();
        MyInterface.staticMethod();
        return CONSTANT;
    }
}

class Example1 {
    public int instanceMethod() {
        MyInterface.staticMethod();
        return MyInterface.CONSTANT;
    }
}
public class Main {
    public static void main(String[] args) throws Throwable {
    }
}

```

var keyword

- Available from JDK10
- used to infer datatype at runtime
- var a=10;
- used only inside constructor or methods or loops or compound block

Rules

1. var cant declare without initial value
2. var can be declared in first line and initialized in second line
3. var cannot be initialized with null value without a type
4. var is not permitted in multiple variable declaration
5. var cannot be used to initialize an array
6. var is reserved type name but not reserved keyword, so we can use var as an identifier except for class, interface ,enum
7. var in lambda expr - you cant mix var and non-var parameters

```

interface MyInterface {
    void add(int a,int b);
}
class Var {
    Var() {
        var var="var"; //correct
    }
    public void var() { //correct
        Var var=new Var(); //correct
    }
}
/*class var{ //error
}
interface var { //error
}
enum var { //error

}*/
public class Main {
    Main(){
        var a1="hello";
    }
    {
        var a2=10;
        a2=11;
    }
}

```

```

public static void main(String[] args) {
    var size=1;
    for(var i=0;i<10;i++) {

    }
    //var x; //error
    var x
        =3.14f;
    var x1=23.34;
    x1=x;
    var y
        ="hello";
    //var z=null; //error
    var z=(String)null;
    var z1="hello";
    z1=null;

    int a=5,b=6;
    // var a1=5,b1=9; //error

    // var a1[]={1,2,4}; //error

    MyInterface m1=(var a1,var b1)->System.out.println(a1+b1); //correct
    MyInterface m2=(int a1,var b1)->System.out.println(a1+b1); //error
}
}
}

```

From: Senthil Kumar T

Sent: Wednesday, September 27, 2023 4:23 PM

To: Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarikumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteeek <akasam.karteeek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen

<sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsriera@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha.m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Sathesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantnrao Gaurkar <rutikahemantnrao.gaurkar@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajeshekethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendrashihare@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujavarapu <priyanka.jujavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Nagar Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premisingh <padya.premisingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgupta@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoovrm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumar@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; Archana Thineta' <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <MonishaI@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyaa <arolla.anuhyaa@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydi@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S

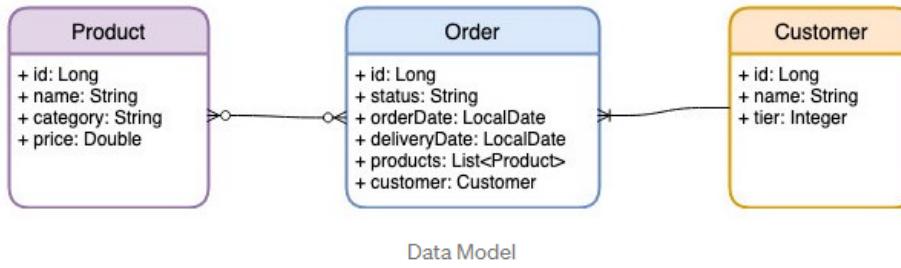
<kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshitijomeses@hcl.com](mailto:bhosalekshitijomeses@hcl.com)>; 'Kshitij Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkatasatya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulassai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraih <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchen Gurunath <kanchen.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; kalpana.golla@hcl.com; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunk@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judson.sam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijau.mallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhima@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitar.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitela@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati.bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <[DHIVYAC@hcl.com](mailto:DHVYAC@hcl.com)>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankilmkar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommaraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 12 Notes

Classification: **Confidential**

Stream API Handson:

Consider given data model,



Use static initializer to create ProductList, OrderList and CustomerList which contains sample Product, Order and Customer

Solve the following questions using Stream API

1. Obtain a list of products belongs to category “Books” with price > 100
2. Obtain a list of order with products belong to category “Baby”
3. Obtain a list of product with category = “Toys” and then apply 10% discount
4. Get the cheapest products of “Books” category
5. Get the 3 most recent placed order
6. Obtain a collection of statistic figures (i.e. sum, average, max, min, count) for all products of category “Books”

Streams API

- contains collection of objects from some sources(Array,List,Set) and process them sequentially
- Collection framework also stores collection of objects and we can manipulate the object, but streams are only used for processing the data and we cant manipulate the data
- present in java.util.stream.* package
- 2 types of streams - finite(fixed number of values) and infinite(unlimited) stream
- 2 types of operation
 1. Intermediate operation - return stream itself - optional operation - we can chain multiple intermediate operation
 - filter(), map(), flatMap(), sorted(), peek(), distinct(), limit(), skip()
 2. Terminal operation - It will traverse into the newly generated stream and return single value - mandatory operation - we cant chain terminal operation
 - toArray(), forEach(), count(), min(), max(), reduce(), collect(), anyMatch(), allMatch(), noneMatch(), findFirst()

3 steps

Creation of Stream - Intermediate operation - Terminal Operation

- If we perform terminal operation then that stream is completely closed, so when we perform any other operation on closed stream then we will get IllegalStateException

1. Creation of Stream - Finite stream - 2 ways

1. stream() - used to generate a stream from some source (ie) array or list or set

```
String s[]={one,two,three};
Stream<String> s1=s.stream();
```

```
List<String> l1=new ArrayList<>();
l1.add("Ram");
l1.add("Sam");
l1.add("Raj");
Stream<String> s2=l1.stream();
```

2. of() - used to create our own stream

```
Stream<Integer> st= Stream.of(1,2,3,4,5);
```

filter() - intermediate - used to filter the data based on some condition

reduce() - terminal - used to group the data into single value

- Optional reduce(BinaryOperator)
- Integer reduce(int initialValue, BinaryOperator)
- Stream reduce(int initialValue, BinaryOperator b1, BinaryOperator b2)

```
public class Main {
    public static void main(String[] args) {
        List<String> l1=new ArrayList<>();
        l1.add("Ram");
```

```

l1.add("Sam");
l1.add("Raj");
l1.add("Tam");
l1.add("Tim");
l1.add("Ram");
l1.add("John");
l1.add("Jack");
System.out.println(l1.size()); //8

Stream<String> s1=l1.stream();
Stream<String> s2=s1.distinct();
long c=s2.count();
System.out.println(c); //7

long c1=l1.stream().distinct().count();
System.out.println(c1); //7

boolean b1=l1.stream().distinct().anyMatch((e)->e.startsWith("R"));
System.out.println(b1); //true

boolean b2=l1.stream().distinct().allMatch((e)->e.startsWith("R"));
System.out.println(b2); //false

boolean b3=l1.stream().distinct().noneMatch((e)->e.startsWith("Z"));
System.out.println(b3); //true

List<Student> l2=new ArrayList<>();
l2.add(new Student(23,"PK"));
l2.add(new Student(26,"KK"));
l2.add(new Student(23,"MK"));
l2.add(new Student(21,"SK"));
l2.add(new Student(40,"RK"));
l2.add(new Student(30,"BK"));
l2.add(new Student(29,"DK"));
l2.add(new Student(28,"GK"));
l2.add(new Student(33,"TK"));

Stream<Student> st=l2.stream().filter((a1)->a1.getId()>25);
st.forEach(System.out::println);

Optional opt=Stream.of(3,5,6).reduce((a,b)->a*b);
System.out.println(opt.get()); //90

Integer i=Stream.of(3,5,6).reduce(2, (a,b)->a*b);
System.out.println(i); //180

Optional<String> opt1=Stream.of("lion","ape","tiger").min((c11,c21)->c11.length()-c21.length());
System.out.println(opt1.get()); //ape
}
}

map() - intermediate
- takes one Function functional interface as an argument and returns a stream consisting of result generated by applying the passed function to each element
- used for data transformation

```

collect() - terminal - used to collect the data as list or set

```

Collectors.toList()
Collectors.toSet()
Collectors.toMap(Function f1,Function f2)
Collectors.joining()
Collectors.counting()
Collectors.toCollection(Supplier s)
Collectors.partitioningBy() - used to split the list into 2 parts based on true and false
- partitioningBy(predicate)
- partitioningBy(predicate,Collector)
Collectors.groupingBy() - group the stream of elements under some condition
- groupingBy(Function)
- groupingBy(Function,Collector)
- groupingBy(Function,Comparator,Collector)

```

flatMap() - intermediate

- takes one Function functional interface as an argument and returns a new stream and that stream is copied to another stream which will return the value
- used for data transformation + flattenning

```
public class Main {
```

```
    public static void main(String[] args) {
        Integer a[] = new Integer[] {1,2,3,4,5};
        List<Integer> l1 = Arrays.asList(a);
```

```
//JDK10
```

```
//List<Integer> l2 = List.of(1,2,3,4,5); //create immutable list
```

```
List<Integer> l2 = l1.stream().map((e) -> e * 3).collect(Collectors.toList());
l2.forEach(System.out::println); //3,6,9,12,15
```

```
List<Integer> l3 = l1.stream().flatMap((e1) -> Stream.of(e1 * 2)).collect(Collectors.toList());
l3.forEach(System.out::println); //2,4,6,8,10
```

```
String s1 = Stream.of("one", "two", "three").collect(Collectors.joining("-"));
System.out.println(s1); //one-two-three
```

```
long c = Stream.of("one", "two", "three").collect(Collectors.counting());
System.out.println(c); //3
```

```
List<String> l4 = Stream.of("lions", "tigers", "bears", "toads", "toads")
    .filter((s) -> s.startsWith("t")).collect(Collectors.toList());
l4.forEach(System.out::println); //tigers,toads,toads
```

```
Set<String> l5 = Stream.of("lions", "tigers", "bears", "toads", "toads")
    .filter((s) -> s.startsWith("t")).collect(Collectors.toSet());
l5.forEach(System.out::println); //tigers,toads
```

```
TreeSet<String> ts = Stream.of("lions", "tigers", "bears", "toads", "toads", "tadpole")
    .filter((s) -> s.startsWith("t")).collect(Collectors.toCollection(TreeSet::new));
System.out.println(ts); // [tadpole,tigers,toads]
```

```
Map<String, Integer> m1 = Stream.of("lions", "tigers", "bears", "toads")
    .collect(Collectors.toMap(k1 -> k1, String::length));
m1.forEach((k, v) -> System.out.println("Key = " + k + " Value = " + v));
```

```
Map<Boolean, List<String>> m2 = Stream.of("lions", "tigers", "bears", "toads", "toads", "tadpole")
    .collect(Collectors.partitioningBy((a1) -> a1.length() <= 5));
System.out.println(m2);
```

```
Map<Boolean, Set<String>> m3 = Stream.of("lions", "tigers", "bears", "toads", "toads", "tadpole")
    .collect(Collectors.partitioningBy((a1) -> a1.length() <= 5, Collectors.toSet()));
System.out.println(m3);
```

```
Map<Integer, List<String>> m4 = Stream.of("lions", "tigers", "bears", "lions", "ape")
    .collect(Collectors.groupingBy((e) -> e.length()));
System.out.println(m4);
```

```
Map<Integer, Set<String>> m5 = Stream.of("lions", "tigers", "bears", "lions", "ape")
    .collect(Collectors.groupingBy((e) -> e.length(), Collectors.toSet()));
System.out.println(m5);
```

```
TreeMap<Integer, Set<String>> m6 = Stream.of("lions", "tigers", "bears", "lions", "ape")
    .collect(Collectors.groupingBy((e) -> e.length(), TreeMap::new, Collectors.toSet()));
System.out.println(m6);
```

```
}
```

```
}
```

sorted() - intermediate - used to sort the elt

```
public class Main {
```

```
    public static void main(String[] args) {
        List<String> l1 = List.of("9", "A", "z", "1", "B", "4", "e", "f");
```

```

List<String> l21=l1.stream().sorted().collect(Collectors.toList()); //asc order
System.out.println(l21);
List<String> l3=l1.stream().sorted(Comparator.reverseOrder()).collect(Collectors.toList()); //desc order
System.out.println(l3);

List<Student> l2=new ArrayList<>();
l2.add(new Student(23,"PK"));
l2.add(new Student(26,"KK"));
l2.add(new Student(23,"MK"));
l2.add(new Student(21,"SK"));
l2.add(new Student(40,"RK"));
l2.add(new Student(30,"BK"));
l2.add(new Student(29,"DK"));
l2.add(new Student(28,"GK"));
l2.add(new Student(33,"TK"));

//comparingInt(), comparingDouble(), comparingLong()
List<Student> l4=l2.stream().sorted(Comparator.comparingInt(Student::getId))
    .collect(Collectors.toList());
l4.forEach(System.out::println);
System.out.println();
List<Student> l5=l2.stream().sorted(Comparator.comparingInt(Student::getId).reversed())
    .collect(Collectors.toList());
l5.forEach(System.out::println);
System.out.println();
List<Student> l6=l2.stream().sorted(Comparator.comparing(Student::getName))
    .collect(Collectors.toList());
l6.forEach(System.out::println);
System.out.println();
List<Student> l7=l2.stream().sorted(Comparator.comparing(Student::getName).reversed())
    .collect(Collectors.toList());
l7.forEach(System.out::println);
}
}

```

Other ways to create stream

1. builder() - create finite stream
2. generate() - create infinite stream
3. iterate() - create infinite stream

Create streams based on primitive datatype

1. IntStream - create stream of int values
2. DoubleStream - create stream of double values
3. LongStream - create stream of long values

From JDK1.9

`takeWhile()` - if stream does not match the predicate, it will discard the rest of stream
`dropWhile()` - if stream does not match the predicate, it will print the rest of stream

```

public class Main {
    public static void main(String[] args) {
        Stream<String> s1=Stream.<String>builder().add("Ram").add("Sam").add("Raj").build();
        s1.forEach(System.out::println);

        Stream<String> s2=Stream.generate(()->"hello").limit(5);
        s2.forEach(System.out::println);

        Stream<Integer> s3=Stream.iterate(2, (i)->i*2).skip(3).limit(5);
        s3.forEach(System.out::println);

        IntStream i1=IntStream.range(1,6); //start to end-1
        i1.forEach(System.out::println); //1 2 3 4 5

        IntStream i2=IntStream.rangeClosed(1,6); //start to end
        i2.forEach(System.out::println); //1 2 3 4 5 6

        IntStream i3="abcd".chars();
        i3.forEach(System.out::println); //97 98 99 100

        Random r=new Random();
        DoubleStream d1=r.doubles(5);
    }
}

```

```

d1.forEach(System.out::println);

List<Student> l2=new ArrayList<>();
l2.add(new Student(23,"PK"));
l2.add(new Student(26,"KK"));
l2.add(new Student(23,"MK"));
l2.add(new Student(21,"SK"));
l2.add(new Student(40,"RK"));
l2.add(new Student(30,"BK"));
l2.add(new Student(29,"DK"));
l2.add(new Student(28,"GK"));
l2.add(new Student(33,"TK"));

IntStream i5=l2.stream().mapToInt(Student::getId);
i5.forEach(System.out::println);

OptionalInt op=l2.stream().mapToInt(Student::getId).max();
System.out.println(op.getAsInt()); //40

OptionalDouble op1=l2.stream().mapToDouble(Student::getId).average();
System.out.println(op1.getAsDouble());

//IntSummaryStatistics,DoubleSummaryStatistics, LongSummaryStatistics
IntSummaryStatistics in=l2.stream().collect(Collectors.summarizingInt(t->t.getId()));
System.out.println(in);
System.out.println(in.getAverage()+" "+in.getCount());

Stream.of(2,4,6,8,9,10,12).takeWhile(n->n%2==0).forEach(System.out::println); //2 4 6 8
Stream.of(2,4,6,8,9,10,12).dropWhile(n->n%2==0).forEach(System.out::println); //9 10 12

}
}

```

From: Senthil Kumar T

Sent: Monday, September 25, 2023 3:29 PM

To: Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.an@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarikumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tava Durga <tava.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarrajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; 'sridhar.at@hcl.com' <sridhar.at@hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep_ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri

Vinod <simhadri.vinod@hcl.com>; Silviu Daniel Antohe Multescu <silviudaniel.antoh@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.mohamed@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandan Neeraja <akula.chand@hcl.com>; Shamugam Samandhvada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyaK@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshis@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepell <kallepell.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; 'ricardo.lopezporras@hcl.com' <ricardo.lopezporras@hcl.com>; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetty Siva Krishna <ramisetty.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; 'rahulvenkat.gone@hcl.com' <rahulvenkat.gone@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavarapu <priyanka.juijavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant.gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premisingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin.sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh.anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoovrm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; 'Nanjula Perera' <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banarker <nandish.banarker@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vitthal Benki <nagarajvitthal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <MonishaI@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyaa <arolla.anuhyaa@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <Ishfaaq.Ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.mohamed@hcl.com>; Metkari Saif Mohammadraphi <metkarisaf.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh

Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekawat@hcl.com>; Lohith N K <lohit.hnk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnapala <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; 'Kshitij Jaiswal' <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha_s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; 'kalpana.golla@hcl.com' <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Parguna <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judson.sam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jjau Mallikarjun Lohar <jjiaumallikar.lohar@hcl.com>; Jelio Halleys Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadarai S <jabadarai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saita@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Varas Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashs@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedula Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznuez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 11 Notes

Classification: **Confidential**

Handson

Exercise 1: Write a lambda expression which accepts x and y numbers and return xy .

Exercise 2: Write a method that uses lambda expression to format a given string, where a space is inserted between each character of string. For ex., if input is "CG", then expected output is "C G".

Exercise 3: Write a method that uses lambda expression to accept username and password and return true or false. (Hint: Use any custom values for username and password for authentication)

Exercise 4: Write a class with main method to demonstrate instance creation using method reference. (Hint: Create any simple class with attributes and getters and setters)

Exercise 5: Write a method to calculate factorial of a number. Test this method using method reference feature.

JDK 1.5

1. var args
2. for each stmt
3. Generics
4. Covariant return type
5. static import
6. Autoboxing and Unboxing
7. Annotation
8. Assertion

JDK1.7

1. In switch stmt, we can pass String as an expr

2. try with resources

3. Underscore literal - used for representation

```
int a=100000;
int a=1_00_000; //correct
int a1=10_000; //correct
int a2=10_; //error
int a3=_10; //error
float f1=10.1_1f; //correct
float f2=10._11f; //error
float f3=10_.11f; //error
float f4=10.11_f; //error
```

4. Binary literal

```
int a=0b11; //3
int a1=0B100; //4
```

5. Multi catch statement using |

6. Generics

```
List<String> l=new ArrayList<>();
```

Java 8

1. Lambda expression

- Used to enable functional programming in Java, until JDK1.7 we cant write anything that exists on its own, first we create a class and access the class with the help of object

```
public class Main {
    public void greet() {
        System.out.println("Hello World");
    }
    public static void main(String[] args) {
        Main m=new Main();
        m.greet();
    }
}
```

Now greet() always print "Hello World", but we need greet() to print different info, until JDK1.7 if we need different implements then we have to go for interface

```
interface Greeting {
    void perform();
}
class HelloWorld implements Greeting {
    public void perform() {
        System.out.println("Helloworld");
    }
}
class Welcome implements Greeting {
    public void perform() {
        System.out.println("Welcome");
    }
}
public class Main {
    /*public void greet() {
        System.out.println("Hello World");
    }*/
    public void greet(Greeting g) {
```

```

        g.perform();
    }
}

public static void main(String[] args) {
    Main m=new Main();
    Greeting gr=new HelloWorld(); //DMD
    gr.perform(); //Helloworld
    gr=new Welcome();
    gr.perform(); //Welcome
}
}

```

We want to pass the behaviour (ie) perform() as an argument to greet() and execute the behaviour directly, rather than passing the thing that contains the behaviour (ie) Greeting interface

Lambda expression are just a function that exists on its own and execute it independently, used to enable functional programming

How to write Lambda expr?

1. We take a function and assign to a variable

```
a=public void perform() {
    System.out.println("Hello");
}
```

-public, private, protected make sense if we write any method inside class, but in lambda expr, functions are exists on its own so when we write lambda expr there is no need to define access specifiers

```
2. a=void perform() {
    System.out.println("Hello");
}
```

Whenever we assign a function to a variable, we can access the function using ur variable name, so when we write lambda expr there is no need to define function name

```
3. a=void () {
    //System.out.println("Hello");
    return "hello";
}
```

By seeing the function itself we can identify the return type of the function, so when we write lambda expr there is no need to define its return type

```
4. a={() {
    System.out.println("Hello");
}}
```

Lambda expr contains parenthesis for input arg, logic and we need to put -> symbol

```
a={() -> {
    System.out.println("Hello");
}}
```

```
b=(int a,int b) -> {
    System.out.println(a+b);
}
```

```
5. FunctionType<Void,Void> a = () -> {
    System.out.println("Hello");
}
```

```
FunctionalInterface a= () -> {
    System.out.println("Hello");
}
```

Use a functional interface as a return type for lambda expr, this concept is called Type Inference

- Lambda expr is always used to logic only for methods in functional interface

Rules

1. If ur body of lambda expr is just a single line, then we can ignore curly braces

```
FunctionalInterface a=() -> System.out.println("hello");
```

2. If ur body of lambda expr is just a single line, then we can ignore the return stmt

```
FunctionalInterface b=(int a) -> {
    return a*2;
}
```

```
FunctionalInterface b=(int a) -> a*2;
```

Before JDK1.8, how we can access methods of any interface ? - 2 ways

1. By implementing interface in a class

```
interface Greeting {
    void perform();
}
class Example implements Greeting {
    @Override
    public void perform() {
        System.out.println("Hello");
    }
}
public class Main {
    public static void main(String[] args) {
        Example e=new Example();
        e.perform();
    }
}
```

2. Using Anonymous Inner class

```
interface Greeting {
    void perform();
}
public class Main {
    public static void main(String[] args) {
        Greeting g=new Greeting() {
            @Override
            public void perform() {
                System.out.println("Heloo");
            }
        };
        g.perform();
    }
}
```

Functional Interface

- contains only one abstract method and any number of default and static method
- We will write logic for functionalinterface using lambda expr
- Using `@FunctionalInterface` annotation - optional - present in `java.lang.*` pkg
- It is also called as SAM(Single Abstract Method)
- If an interface contains any method of Object class as abstract, then it is also considered as functional interface

```
//Functional interface
interface A {
    void add();
}
```

```
//Functional interface
interface A {
    void add();
    default void show() {
    }
    static void show1() {
    }
}
```

```
//Not Functional interface
interface A {
    void add();
    int add1();
}
```

```
//Functional interface
interface A {
    void add();
    String toString(); //Object class
}
```

```

//Not Functional interface
interface A {
    void add();
    boolean equals();
}

//Functional interface
interface A {
    void add();
    boolean equals(Object o); //Object class
}

//Functional interface
@FunctionalInterface
interface A {
    void add();
}

//Functional interface
interface A {
    void add();
}

//Not Functional interface
interface B extends A {
    void add1();
}

interface Greeting {
    void perform();
}

public class Main {
    public static void main(String[] args) {
        Greeting gr=()->System.out.println("Using lambda");
        gr.perform(); //Using lambda
    }
}

```



```

interface MyInterface {
    int calculateLength(String s);
}

public class Main {
    public static void main(String[] args) {
        MyInterface m1=(e)->e.length();
        System.out.println(m1.calculateLength("hello")); //5
    }
}

```



```

interface MyInterface {
    void add(int a,int b);
}

public class Main {
    public static void main(String[] args) {
        MyInterface m1=(x,y)->System.out.println(x+y);
        m1.add(2, 3); //5
    }
}

```

2. Method Reference

- also used to access the method of functional interface, instead of writing logic separately using lambda expr, we can refer already existing method, when the signature of interface method is same as static method or instance method or constructor

- Alternative to Lambda expr
- Denoted by ::

3 types

1. Reference to static method
2. Reference to instance method

```
interface MyInterface {
```

```

        void show();
    }
public class Main {
    /*public static void add() {
        System.out.println("Using reference to static method ");
    }*/
    public void add1() {
        System.out.println("Using reference to instance method ");
    }
public static void main(String[] args) {
    // MyInterface m1=()->System.out.println("Using Lambda");
    // m1.show(); //Using Lambda

    //MyInterface m2=Main::add;
    //m2.show(); //Using reference to static method

    Main m3=new Main();
    MyInterface m4=m3::add1;
    m4.show();
}
}

```

3. Reference to a constructor

```

interface MyInterface {
    //Main show();
    Main show(String s);
}
public class Main {
    Main(){
        System.out.println("Default constructor");
    }
    Main(String s){
        System.out.println("Parameterized constructor "+s);
    }
public static void main(String[] args) {
    //MyInterface m1=Main::new;
    //m1.show();
    MyInterface m2=Main::new;
    m2.show("hello");
}
}

```

Predefined Functional Interface - present in java.util.function.* pkg

1. Consumer interface
- void accept(T t)

JDK1.8 - void forEach(Consumer) - used to iterate the elts of List and Set interface individually

2. BiConsumer interface
- void accept(T t, U u)

JDK1.8 - void forEach(BiConsumer) - used to iterate the elts of Map interface individually

3. Function interface
- R apply(T t)

4. BiFunction interface
- R apply(T t,U u)

5. Predicate interface
- boolean test(T t)

6. BiPredicate interface
- boolean test(T t,U u)

7. Supplier interface
- T get()

8. BooleanSupplier interface

- boolean getAsBoolean()

9. BinaryOperator interface extends BiFunction interface

- R apply(T t,U u)

10. UnaryOperator interface extends Function interface

- R apply(T t)

From: Senthil Kumar T

Sent: Friday, September 22, 2023 3:27 PM

To: Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopal Reddy Koppula <venugopalar.koppu@hcl.com>; Venkatesh S <venkatesh.s@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Venam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Yamshi Krishna Bitla <yamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.an@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalaran.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; 'Troshan Mihindu' <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvvu Swapna <duvvu.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeeli Chandran <siranjeeli.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silviu Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh.a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashank Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha.m@hcl.com>; Saubhagyaa Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakhija <sanidhya.sakhija@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer.s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhone <sakshi.dhone@hcl.com>; Amit Kashyap <amit.kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shaminii <shaminii.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepell <kallepell.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan.pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddii <ramu.doddii@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra

<rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andhareartibaban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulverkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavarapu <priyanka.juijavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant.gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornaya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanmant@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vattcharla <vattcharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagaraj.vittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monisha@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyaa <arolla.anuhyaa@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaf.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonza@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапala <[ashini.rathnапала@hcl.com](mailto:ashini.rathnапala@hcl.com)>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshitij.somes@hcl.com](mailto:bhosalekshitij.somes@hcl.com)>; 'Kshitiji Jaiswal' <kshitiji.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <athieeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha.s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiha <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venka Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumatr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchen Gurunath <kanchen.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; Kalpana Golla <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunk@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judson.sam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijau.mallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>;

Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit.mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hpurasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitaja@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; 'Mate Fanindra Khushal' <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <[DHIVYAC@hcl.com](mailto:DHVYAC@hcl.com)>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <oopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvi Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brinda Vasudevan <brinda.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedula Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 10 Notes

Classification: **Confidential**

Handson

1.TreeMap-Player Details

BCCI, for the upcoming IPL season in 2017 decided to give unique cap numbers to every player. Player capNumber is a string. The capNumber and player details are stored in a Treemap.

Create a class named Player with the following private attributes --- name,team and skill. Create a list of objects of Player type.

Cap number is the key and player details is the value. Write a program to display the details of all the players stored in this TreeMap.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output:

Enter the number of players

2

Enter the details of the player 1

57

Jaspirit Bumrah

Mumbai Indians

Bowler

Enter the details of the player 2

55

MS Dhoni

Rising Pune Supergiants

All Rounder

Player Details

55--MS Dhoni--Rising Pune Supergiants--All Rounder

57--Jaspirit Bumrah--Mumbai Indians--Bowler

2.TreeMap-Player Details

BCCI, for the upcoming IPL season in 2017 decided to give unique cap numbers to every player. Player capNumber is a string. The capNumber and player details are stored in a Treemap.

Create a class named Player with the following private attributes --- name,team and skill. Create a list of objects of Player type.

Cap number is the key and player details is the value. Write a program to display the details of all the players stored in this TreeMap and details of the players by searching based on cap number.

Input and Output Format:

Refer sample input and output for formatting specifications.
All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output 1:

```
Enter the number of players
2
Enter the details of the player 1
57
Jaspirit Bumrah
Mumbai Indians
Bowler
Enter the details of the player 2
55
MS Dhoni
Rising Pune Supergiants
All Rounder
Player Details
55--MS Dhoni--Rising Pune Supergiants--All Rounder
57--Jaspirit Bumrah--Mumbai Indians--Bowler
Enter the cap number of the player to be searched
55
Player Details
MS Dhoni--Rising Pune Supergiants--All Rounder
```

Sample Input and Output 2:

```
Enter the number of players
2
Enter the details of the player 1
57
Jaspirit Bumrah
Mumbai Indians
Bowler
Enter the details of the player 2
55
MS Dhoni
Rising Pune Supergiants
All Rounder
Player Details
55--MS Dhoni--Rising Pune Supergiants--All Rounder
57--Jaspirit Bumrah--Mumbai Indians--Bowler
Enter the cap number of the player to be searched
34
Player Details
Player not found
```

Map interface

- used to store collection of object but in the form of unique key value pair
- Map is unordered

Methods

1. void put(Object k, Object v) - insert single key value pair
2. void putAll(Map m) - store multiple key value pair
3. Object putIfAbsent(Object k, Object v) - if specified key is not associated with the value then it associates the key with given value and returns null
4. boolean containsKey(Object k)
5. boolean containsValue(Object v)
6. Object get(Object k) - if we provide the key, it will return value associated with that key
7. Object getOrDefault(Object k, Object defaultValue)
8. Object remove(Object k) - if we provide the key, it will remove value associated with that key
9. boolean remove(Object k, Object v)
10. Object replace(Object k, Object v)
11. boolean replace(Object k, Object oldval, Object newval)
12. boolean isEmpty()
13. int size()

We cant apply Iterator interface directly on Map interface, we have to convert Map to Set interface
14. Set entrySet() - convert both key and value to Set interface

15. Set keySet() - convert only keys to Set interface

Map.Entry interface

- used to describe key and value separately

Methods

1. Object getKey()
2. Object getValue()
3. Object setValue(Object k)

SortedMap interface

- used to sort the elements of Map interface

Methods

1. Object firstKey()
2. Object lastKey()
3. SortedMap subMap(Object start, Object end) - start to end-1
4. SortedMap headMap(Object k)
5. SortedMap tailMap(Object k)

3 classes

1. HashMap class

- uses hashing technique to print the value in random order and contains key value pairs
- default capacity is 16
- allow one null key allowed and any number of null values

Syntax: public class HashMap extends AbstractMap implements Map, Cloneable, Serializable

Constructor

1. HashMap()
2. HashMap(int capacity)
3. HashMap(int capacity, float fillratio)
4. HashMap(Map m)

```
Map<Employee,String> hm=new HashMap<>();
```

So when we create an object for HashMap, internally it creates a bucket from 0 to 15. Each bucket internally considered as LinkedList, this LinkedList contains one node, the internal structure of node will have key,value,hashcode,next

```
Employee e1=new Employee(1,"Ram");
Employee e2=new Employee(2,"Sam");
Employee e3=new Employee(3,"Tam");
Employee e4=new Employee(4,"Jam");
```

```
hm.put(e1,"HR");
```

So when we call put(), first it calculate hashcode for the key using hashCode() (ie) hashCode(e1) = 2000
 Next it will calculate the index using index=hashcode(key) & (n-1)
 $=2000 \& (16-1) = 2000 \& 15 = 6$

it stores e1 in index 6

```
hm.put(e2,"Operation");
```

So when we call put(), first it calculate hashcode for the key using hashCode() (ie) hashCode(e2) = 1000
 Next it will calculate the index using index=hashcode(key) & (n-1)
 $=1000 \& (16-1) = 1000 \& 15 = 9$

it stores e2 in index 9

```
hm.put(e3,"Admin");
```

So when we call put(), first it calculate hashcode for the key using hashCode() (ie) hashCode(e3) = 2000
 Next it will calculate the index using index=hashcode(key) & (n-1)
 $=2000 \& (16-1) = 2000 \& 15 = 6$

it stores e3 in index 6

If same bucket contains multiple node then that is called as Hashing collision, so map will not directly add the entry into the bucket, since both key have same index, map internally calls equals() and check whether the content is same or different, if it is different it will store the entry into same bucket and if it is same it will replace the entry

```
hm.put(null,"IT");
```

If we pass key as null, then entry will be directly stored inside 0th bucket

How searching works in LinkedList?

item1:P->item2:P->item3:P->item4:P

get() of HashMap will return the value very faster

when we call hm.get(key), first JVM will find hashCode(key) and from hashCode we can find the bucket index, so JVM will directly go to the index and find the value, so we use Hashmap because of time complexity

In general, HashMap index can contain multiple nodes in single bucket, ideally have only one node in one bucket. In hash collision, JVM will traverse one by one to fetch the elt, so performance is slow

From Java8, HashMap has been enhanced to reduce the time when using get()

```
Map<Integer, String> hm1 = new HashMap<>();
hm1.put(4, "a");
hm1.put(6, "a");
hm1.put(2, "a");
hm1.put(7, "a");
hm1.put(1, "a");
hm1.put(5, "a");
hm1.put(3, "c");
```

Consider all keys have same hashCode so all will have same index in bucket, so performance slow

In Java8, it will convert LinkedList into tree structure, when no of node reaches certain threshold and that threshold is called Treeify threshold

```
Map<Integer, Integer> hm1 = new HashMap<>();
hm1.put(4, 4);
hm1.put(6, 6);
hm1.put(2, 2);
hm1.put(7, 7);
hm1.put(1, 1);
hm1.put(5, 5);
hm1.put(3, 3);
```

Hence we store integer it takes bigger or smaller, in case of String or object values, so JVM will determine that particular object is greater or smaller using compareTo()

```
public class Main {
```

```
    public static void main(String[] args) {
        Map<Integer, String> hm = new HashMap<>();
        System.out.println(hm.size()); //0
        hm.put(10, "apple");
        hm.put(13, "grapes");
        hm.put(15, "banana");
        hm.put(16, "Guava");
        hm.put(10, "mango"); //no error, the value will be overridden
        System.out.println(hm.size()); //4
        System.out.println(hm); //print in curly braces, in random order

        System.out.println(hm.putIfAbsent(20, "Melon")); //null
        System.out.println(hm.putIfAbsent(11, "Berry")); //null
        System.out.println(hm.putIfAbsent(10, "Lime")); //mango
        System.out.println(hm.size()); //6
        System.out.println(hm.get(16)); //Guava
        System.out.println(hm.get(17)); //null
        System.out.println(hm.getOrDefault(17, "Pineapple")); //Pineapple
        System.out.println(hm.size()); //6
        hm.remove(10);
        System.out.println(hm.get(10)); //null
        System.out.println(hm.remove(11, "Berry")); //true
        System.out.println(hm.get(11)); //null
        System.out.println(hm.remove(13, "Berry")); //false
        System.out.println(hm.get(13)); //grapes

        System.out.println(hm.replace(16, "Guava", "Orange")); //true
        System.out.println(hm.get(16)); //Orange
        System.out.println(hm.replace(20, "Greenapple")); //Melon
        System.out.println(hm.get(20)); //Greenapple
        System.out.println(hm);

        Set set = hm.entrySet(); //convert both key and value to Set intf
        Iterator itr = set.iterator();
```

```

        while(itr.hasNext()) {
            Map.Entry me=(Map.Entry)itr.next(); //one key value pair
            System.out.println(me.getKey()+" "+me.getValue());
        }

        set=hm.keySet(); //convert only key to Set intf
        Iterator<Integer> itr1=set.iterator();
        while(itr1.hasNext()) {
            Integer key=itr1.next(); //one key
            System.out.println(key+" "+hm.get(key));
        }
    }
}

```

2. LinkedHashMap class
- maintain the insertion order

Syntax: public class LinkedHashMap extends HashMap

Constructor

1. LinkedHashMap()
2. LinkedHashMap(int capacity)
3. LinkedHashMap(int capacity, float fillratio)
4. LinkedHashMap(Map m)

3. TreeMap class
- used to sort the elements only based on key
- u cant add null key, we can have null values

Syntax: public class TreeMap extends AbstractMap implements NavigableMap, Cloneable, Serializable

4. HashTable class
- It is a legacy class
- similar to HashMap but it is synchronized or threadsafe

Syntax: public class Hashtable extends Dictionary implements Map, Cloneable, Serializable

Constructor

1. Hashtable()
2. Hashtable(int capacity)
3. Hashtable(int capacity, float fillratio)
4. Hashtable(Map m)

5. Properties class
- contains key value pairs but both key and value should be in the form of String, print in random order

Syntax: public class Properties extends Hashtable

Constructor

- 1.Properties()
- 2.Properties(String default)

public class Main {

```

    public static void main(String[] args) {
        LinkedHashMap<Integer,String> hm=new LinkedHashMap<>();
        System.out.println(hm.size()); //0
        hm.put(10, "apple");
        hm.put(13, "grapes");
        hm.put(15, "banana");
        hm.put(16, "Guava");
        hm.put(10, "mango"); //no error, the value will be overridden
        System.out.println(hm);

        TreeMap<String,Integer> tm=new TreeMap<>();
        tm.put("lion",10);
        tm.put("cat",30);
        tm.put("ape",20);
        tm.put("bear",40);
        System.out.println(tm);
    }
}

```

```

Hashtable<Integer,Double> ht=new Hashtable<>();
System.out.println(ht.size());
ht.put(3, 2.45);
ht.put(1, 5.67);
ht.put(6, 4.23);
ht.put(7, 1.23);
System.out.println(ht);

Enumeration<Integer> e=ht.keys();
while(e.hasMoreElements()) {
    Integer key=e.nextElement();
    System.out.println(key+" "+ht.get(key));
}

Properties p=new Properties();
p.put("cat", "milk");
p.put("dog", "meat");
p.put("cow", "grass");
p.put("fish", "water");
System.out.println(p);

Set set=p.keySet(); //only key
Iterator<String> itr=set.iterator();
while(itr.hasNext()) {
    String key=itr.next();
    System.out.println(key+" "+p.getProperty(key));
}
}
}
}

```

From: Senthil Kumar T

Sent: Monday, September 18, 2023 3:24 PM

To: Yerramsetti V Subrahmanya Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.an@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan.s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <swarje.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarrajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra.s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha.rani@hcl.com>; Aher Rutuja Rajedra <aherrutuja.rajedra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteeck <akasam.karteeck@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudelian.anto@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>;

Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant.shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyK@hcl.com>; Shailesh Kumar Yadav <shaileshkumaryadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsrierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha.m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakhuja <sanidhya.sakhuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhone <sakshi.dhone@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepal <rajesh.kethepal@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujavarapu <priyanka.jujavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khatti <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premisingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevane.mondh@hcl.com>; Aravind V <aravind_v@hcl.com>; Nagaraj Vittal Benki <nagaraj.vittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth_l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; Archana Thineta <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakcharla Mounika <vakcharla.sri@hcl.com>; Monisha. I <MonishaI@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Ishfaaq.Ismath <Ishfaaq.Ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.mohamed@hcl.com>; Metkari Saif Mohammadadrapi <metkarisaf.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnapala <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu

Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritisak@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha_s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumarr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.maland@hcl.com>; kalpana.golla@hcl.com; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadarai S <jabadarai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemant K <hemant-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Gunduru Dheekshitha <gunduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saita@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <[DHIVYAC@hcl.com](mailto:DHVYAC@hcl.com)>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brinda Vasudevan <brinda.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedula Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gavararammana <chathuraruwan.gavar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 9 Notes

Classification: **Confidential**

Handson

1. Write a Java program to read all the player information from the user and display the player name and their cap number sorted based on their cap number (descending order). The contact details consist of player name, skill and cap number. Use Collections.sort() method for sorting.

Create a main class "Main.java"

Create Player class with below private attributes

playerName - String

skill - String

capNumber - Long

Add appropriate getter and setter methods for Player class

Include a constructor for Player class with the arguments player name, skill and cap number

Implement Comparable interface and implement the method compareTo() to perform sorting based on cap number

Input and Output Format:

First input corresponds to the number of players and followed by each player's information.

Refer sample input and output for formatting specifications.

[All text in bold corresponds to input and the rest corresponds to output]

Sample Input/Output :

Enter number of players:

2

Enter player 1 detail

Enter Name

Suresh Raina

Enter Skill

Batsman

Enter Cap Number

265

Enter player 2 detail

Enter Name

Virat Kohli

Enter Skill

Batsman

Enter Cap Number

268

Player list after sorting by cap number in descending order

Virat Kohli-268

Suresh Raina-265

2.Comparator - no of matches player

Write a Java program to get the team name and number of matches played by the team from the user and display a report with team name and number of matches sorted based on the number of matches in ascending order. Use Collection.sort() method to perform the sorting in your main class. Send the Comparator object as second argument to the sort method to use this comparator for sorting.

Create a main class " Main.java"

Create a class named Team with the following private member variables / attributes,

name - String

numberOfMatches - Long

Include a constructor accepting Team name and number of matches as arguments

Add appropriate getter and setter methods for Team class

Create TeamComparator implementing Comparator interface

Implement compare method to compare two team objects based on their number of matches played.

Input and Output Format:

First input corresponds to the number of teams and followed by each team information.

Refer sample input and output for formatting specifications.

[All text in bold corresponds to input and the rest corresponds to output]

Sample Input/Output :

Enter number of teams:

3

Enter team 1 detail

Enter Name

Chennai super Kings

Enter number of matches

132

Enter team 2 detail

Enter Name

Royal Challengers Bangalore

Enter number of matches

139

Enter team 3 detail

Enter Name

Delhi Daredevils

```

Enter number of matches
131
Team list after sort by number of matches
131
132
139

```

Collections class
- contain static methods/algorithms that support util package

Methods

1. static int binarySearch(List l,int val) - search any element in list
2. static void copy(List dest,List src)
3. static List nCopies(int val, Object o) - create an immutable list with list of values
4. static boolean disjoint(List l1, List l2) - return true if there is no common elt
5. static List emptyList() - create an immutable list
6. static Set emptySet() - create an immutable set
7. static Map emptyMap() - create an immutable map
8. static void fill(List l, Object o) - fill the list with particular value
9. static int frequency(List l, Object o) - return how many times an elt occur in the list
10. static boolean replaceAll(List l, Object oldval, Object newval)
11. static Object max(List l) - find max elt
12. static Object min(List l) - find min elt
13. static void reverse(List l)
14. static void shuffle(List l) - randomly shuffle elt
15. static List singletonList(Object o) - create list with single elt
16. static Set singleton(Object o)
17. static Map singletonMap(Object k, Object v)
18. static void sort(List l)
19. static void sort(List l, Comparator c)
20. static void swap(List l, int oldindex, int newindex)
21. static List synchronizedList(List l)
22. static Set synchronizedSet(Set s)
23. static Map synchronizedMap(Map m)
24. static List unmodifiableList(List l)
25. static Set unmodifiableSet(Set s)
26. static Map unmodifiableMap(Map m)

```

public class Main {
    public static void main(String...args) {
        List<Integer> l1=new ArrayList<>();
        l1.add(3);
        l1.add(5);
        l1.add(6);
        l1.add(7);
        l1.add(9);
        System.out.println(Collections.binarySearch(l1, 6)); //2

        List<String> l2=new ArrayList<>();
        l2.add("one");
        l2.add("two");
        l2.add("three");
        List<String> l3=new ArrayList<>();
        l3.add("four");
        l3.add("five");
        l3.add("six");
        //l3.add("seven");

        System.out.println(Collections.disjoint(l2, l3)); //true
        Collections.copy(l3, l2);
        System.out.println(l2); // [one,two,three]
        System.out.println(l3); // [one,two,three]
        System.out.println(Collections.disjoint(l2, l3)); //false

        List l4=Collections.nCopies(3, "Hello");
        System.out.println(l4); // [Hello,Hello,Hello]
        //l4.add("Hi");
        //System.out.println(l4); //UnsupportedOperationException
    }
}

```

```
List l5= Collections.emptyList();
System.out.println(l5); //[]
//l5.add("Hi");
// System.out.println(l5); //exception

System.out.println(Collections.frequency(l4, "Hello")); //3
Collections.fill(l3, "Hi");
System.out.println(l3); // [Hi,Hi,Hi]
Collections.replaceAll(l3, "Hi", "He");
System.out.println(l3); // [He,He,He]

List<String> l6= new ArrayList<>();
l6.add("Hi!");
l6.add("Hi");
l6.add("Hiiii");
l6.add("Hi Good monring");
Collections.replaceAll(l6, "Hi", "He");
System.out.println(l6); // [Hii,He,Hiiii,Hi Good morning]

List<Integer> l7= new ArrayList<>();
l7.add(8);
l7.add(-20);
l7.add(-8);
l7.add(20);
System.out.println(l7); // [8,-20,-8,20]

Collections.reverse(l7);
System.out.println(l7); // [20,-8,-20,8]
System.out.println(Collections.max(l7)); //20
System.out.println(Collections.min(l7)); //-20

Collections.shuffle(l7);
System.out.println(l7); //random order
Collections.sort(l7); //ascending order
System.out.println(l7); // [-20,-8,8,20]

Collections.sort(l7, Comparator.reverseOrder()); //descending order
System.out.println(l7); // [20,8,-8,-20]

Collections.swap(l7, 0, 3);
System.out.println(l7); // [20,-8,8,-20]

l7= Collections.singletonList(23);
System.out.println(l7); // [23]
//l7.add(24);
// System.out.println(l7); //exception

List l8= Collections.synchronizedList(l6);
System.out.println(l8); // threadsafe

List l9= Collections.unmodifiableList(l2);
System.out.println(l9); // [one,two,three] - immutable
l9.add("Hello");
System.out.println(l9); //exception
}
```

Arrays class

- used to perform operation on any datatype array

Methods

- 1. static List asList(int[] a) - convert int array to List
 - 2. static void fill(int[] a, int val) - fill all elts of array with val
 - 3. static void fill(int[] a,int start,int end,int val) - fill elt with val from start to end-1
 - 4. static void sort(int[] a) - sort all elts of array
 - 5. static void sort(int[] a, int start,int end) - sort elts from start to end
 - 6. static boolean equals(int[] a1,int[] a2)
 - 7. static int binarySearch(int[] a,int val)

```
public class Main {  
    public static void main(String...args) {
```

```

Integer a[] = new Integer[10];
for(int i=0;i<10;i++) {
    a[i]=i*-3;
}
for(int a1:a)
    System.out.println(a1);

List<Integer> l=Arrays.asList(a);
System.out.println(l); // [0,-3,-6,-9,-12,-15,-18,-21,-24,-27]

Arrays.sort(a); //[-27,-24,-21,-18,-15,-12,-9,-6,-3,0]
System.out.println(Arrays.binarySearch(a, -15)); //4

//If value is not present it returns negative value based on -(index)-1
//[-27,-24,-21,-18,-15,-12,-10,-9,-6,-3,0] = -(6)-1
System.out.println(Arrays.binarySearch(a, -10)); // -7
//[-27,-24,-22,-21,-18,-15,-12,-10,-9,-6,-3,0] = -(2)-1
System.out.println(Arrays.binarySearch(a, -22)); // -3

Arrays.sort(a); //[-27,-24,-21,-18,-15,-12,-9,-6,-3,0]
Arrays.fill(a, 2, 5, -1); //start to end-1 by -1
//[-27,-24,-1,-1,-1,-12,-9,-6,-3,0]
Arrays.sort(a, 2, 5); //[-27,-24,-12,-1,-1,-1,-9,-6,-3,0]
Arrays.sort(a); //[-27,-24,-12,-9,-6,-3,-1,-1,-1,0]
}

}

```

- If we want to sort the elts of List interface, we have to use Collections.sort(List l)
- If we want to sort the elts of Set interface, we have to use TreeSet class
- If it is numbers then it will sorted in ascending order and String means it will sorted in alphabetical order

Comparable and Comparator interface

- If ur List or Set interface contains collection of user defined object like Employee, Student, Player etc, now we want to sort based on their properties rather than sorting its reference

```
List<Employee> l=new ArrayList<>();
empid,ename,salary,age - properties
```

Comparable interface	Comparator interface
1. provides single sorting sequence (ie) we can do sorting based on only property either by empid or ename or salary or age	1. provides multiple sorting sequence (ie) we can do sorting based on empid and name and salary age
2. present in java.lang.*	2. present in java.util.*
3. int compareTo(Object o)	3. int compare(Object o1, Object o2)
4. We do sorting using Collections.sort(List l) for List intf and TreeSet class for Set intf	4. We do sorting using Collections.sort(List l, Comparator c) for List intf and TreeSet(Comparator c) for Set intf

```

public class Student implements Comparable<Student> {
    private Integer stuld;
    private String name;
    private Integer age;
    public Integer getStuld() {
        return stuld;
    }
    public void setStuld(Integer stuld) {
        this.stuld = stuld;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {

```

```
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    public Student(Integer stuld, String name, Integer age) {
        super();
        this.stuld = stuld;
        this.name = name;
        this.age = age;
    }
    public Student() {
        super();
        // TODO Auto-generated constructor stub
    }
    @Override
    public int compareTo(Student o) {
        // TODO Auto-generated method stub
        return name.compareTo(o.name);
    }
    /*@Override
    public int compareTo(Student o) {
        if(age==o.age)
            return 0;
        else if(age>o.age)
            return 1;
        else
            return -1;
    }*/
    @Override
    public String toString() {
        return "Student [stuld=" + stuld + ", name=" + name + ", age=" + age + "]";
    }
}

public class Employee {
    Integer eld;
    String name;
    Integer age;
    public Integer getEld() {
        return eld;
    }
    public void setEld(Integer eld) {
        this.eld = eld;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }
    public Employee(Integer eld, String name, Integer age) {
        super();
        this.eld = eld;
        this.name = name;
        this.age = age;
    }
    public Employee() {
        super();
        // TODO Auto-generated constructor stub
    }
    @Override
    public String toString() {
```

```
        return "Employee [id=" + id + ", name=" + name + ", age=" + age + "]";
    }
```

```
}
```

```
public class AgeComparator implements Comparator<Employee> {
```

```
    @Override
    public int compare(Employee o1, Employee o2) {
        if(o1.age==o2.age)
            return 0;
        else if(o1.age>o2.age)
            return 1;
        else
            return -1;
    }
}
```

```
}
```

```
public class NameComparator implements Comparator<Employee>{
```

```
    @Override
    public int compare(Employee o1, Employee o2) {
        return o1.name.compareTo(o2.name);
    }
}
```

```
}
```

```
public class Main {
```

```
    public static void main(String...args) {
        List<Integer> l1=new ArrayList<>();
        l1.add(4);
        l1.add(2);
        l1.add(5);
        l1.add(1);
        l1.add(6);
        System.out.println(l1); // [4,2,5,1,6]
        Collections.sort(l1);
        System.out.println(l1); // [1,2,4,5,6]
        Collections.sort(l1,Comparator.reverseOrder());
        System.out.println(l1); // [6,5,4,2,1]
```

```
TreeSet<String> ts=new TreeSet<>();
ts.add("H");
ts.add("B");
ts.add("E");
ts.add("A");
System.out.println(ts); // [A,B,E,H]
TreeSet ts1=(TreeSet) ts.descendingSet();
System.out.println(ts1); // [H,E,B,A]
```

```
List<Student> l2=new ArrayList<>();
l2.add(new Student(103,"Ram",24));
l2.add(new Student(101,"Sam",20));
l2.add(new Student(104,"Raj",21));
l2.add(new Student(100,"Tam",23));
l2.add(new Student(102,"Jam",19));
```

```
Collections.sort(l2); // internally calls compareTo
for(Student s1:l2)
    System.out.println(s1);
```

```
System.out.println();
TreeSet<Student> ts2=new TreeSet<>();
ts2.add(new Student(103,"Ram",24));
ts2.add(new Student(101,"Sam",20));
ts2.add(new Student(104,"Raj",21));
```

```

ts2.add(new Student(100,"Tam",23));
ts2.add(new Student(102,"Jam",19));
for(Student s1:ts2)
    System.out.println(s1);

List<Employee> l3=new ArrayList<>();
l3.add(new Employee(103,"Ram",24));
l3.add(new Employee(101,"Sam",20));
l3.add(new Employee(104,"Raj",21));
l3.add(new Employee(100,"Tam",23));
l3.add(new Employee(102,"Jam",19));

Collections.sort(l3,new AgeComparator());
for(Employee e1:l3)
    System.out.println(e1.getAge());

System.out.println();
Collections.sort(l3,new NameComparator());
for(Employee e1:l3)
    System.out.println(e1.getName());

System.out.println();
TreeSet<Employee> ts3=new TreeSet<>(new AgeComparator());
ts3.add(new Employee(103,"Ram",24));
ts3.add(new Employee(101,"Sam",20));
ts3.add(new Employee(104,"Raj",21));
ts3.add(new Employee(100,"Tam",23));
ts3.add(new Employee(102,"Jam",19));
for(Employee e1:ts3)
    System.out.println(e1);
}
}

```

Map interface

- used to store collection of object but in the form of unique key value pair
- Map is unordered

Methods

1. void put(Object k,Object v) - insert single key value pair
2. void putAll(Map m) - store multiple key value pair
3. Object putIfAbsent(Object k,Object v) - if specified key is not associated with the value then it associates the key with given value and returns null
4. boolean containsKey(Object k)
5. boolean containsValue(Object v)
6. Object get(Object k) - if we provide the key, it will return value associated with that key
7. Object getOrDefault(Object k,Object defaultValue)
8. Object remove(Object k) - if we provide the key, it will remove value associated with that key
9. boolean remove(Object k,Object v)
10. Object replace(Object k,Object v)
11. boolean replace(Object k, Object oldval, Object newval)
12. boolean isEmpty()
13. int size()
14. Set entrySet()
15. Set keySet()

Map.Entry interface

- used to describe key and value separately

Methods

1. Object getKey()
2. Object getValue()
3. Object setValue(Object k)

SortedMap interface

- used to sort the elements of Map interface

Methods

1. Object firstKey()
2. Object lastKey()
3. SortedMap subMap(Object start, Object end) - start to end-1
4. SortedMap headMap(Object k)
5. SortedMap tailMap(Object k)

From: Senthil Kumar T
Sent: Friday, September 15, 2023 3:26 PM
To: Yerramsetti V Subrahmanyamaisatyanarayana <yerramsetti.vsubra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Balu Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.an@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthan <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tun거나 Vishnu Vardhan Rao <tunenga.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajednra <aherrutuja.rajednra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhashis Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at <sridhar.at@hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; 'Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashank Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellulla <shaikibrahim.khale@hcl.com>; Shaik Afried <shaik.afried@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settippalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagyam Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghu <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakuja <sanidhya.sakuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepell <kallepell.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwananaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vdayana <ravikumar.vdayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetty Siva Krishna <ramisetty.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andhareartibaban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahul.sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavavarpu <priyanka.juijavavarpu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>;

Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanmant@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorvm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <MonishaI@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhya <arolla.anuhya@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.aggarwal@hcl.com>; M Hemalatha <hemalatha@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапала <ashini.rathnапала@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshitji.somes@hcl.com](mailto:bhosalekshitji.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondouj Srinidhi <kondouj.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollienkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakam Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpama Saraiha <kasipatasamarpama.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venka Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil.kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; Kalpana Golla <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunam <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijau.mallikarjun.lohar@hcl.com>; Jelio Hallies Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramnani@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dinesh@hcl.com>; Guduru Dheekshitha

<guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitela@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendaran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david.lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommura@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 8 Notes

Classification: Confidential

Handson

1. The Chennai Super Kings were the most successful team in the IPL with a win percentage of 60.68 and had won the title twice in succession (2010 and 2011).

The CSKs have played “n” matches so far in IPLs. Given the number of matches “n” that CSK has played and their team score in the matches as a list, write a program to find the total runs and the average runs scored by the team in all “n” matches.

Input Format:

First line of the input is an integer “n” that corresponds to the number of matches played by CSK.

Next “n” lines contains an integer in each line, that corresponds to the runs scored by CSK in each of the “n” matches.

Output Format:

Output should print in the first line the integer that gives the total runs scored by CSK in the matches.

In the second line, print a float value that gives the average runs.

Sample Input :

```
5
200
210
180
176
192
```

Sample Output :

```
958
191.6
```

2. A measure of a batsman's greatness is his ability to score runs on foreign conditions. It is quite obvious in Cricket that most batsmen have been excellent at home grounds but flops overseas. Consequently not many teams have aggregates balanced in terms of home and away performances.

Sunil now wanted to analyze the performance of IPL teams based on the runs scored in home as well away matches. Given are the team name, number of matches played by the team in home ground “n” and away grounds “m” respectively, runs scored by the team in each of the matches both home and away respectively. Write a program to store the runs scored by the team in both home ground and in other grounds in a list and help Sunil to display the score (in both home and away grounds) of the team that is greater than 300.

Input Format:

First line of the input contains a string that gives the name of the IPL team.

Second line of input contains the integer “n” that corresponds to the number of matches played by the team in home grounds.

Next "n" lines contains the runs scored by the team in each of the matches in home grounds.

Next line that follows contains the integer "m" that corresponds to the number of matches played by the team in away grounds.

Next "m" lines contains the runs scored by the team in each of the matches in away grounds.

Output Format:

Output should display the runs scored by the team in both home ground and in other grounds as a list, line after line.

In the lines to follow, the output should display the score (in both home and away grounds) of the team that is greater than 300, line after line.

Sample Input and output :

```
Enter the team name
Chennai Super Kings
Enter the number of matches played in home ground
2
Enter the runs scored
290
320
Enter the number of matches played in other ground
3
Enter the runs scored
399
180
150
Runs scored by Chennai Super Kings
290
320
399
180
150
Run scored by Chennai Super Kings more than 300
320
399
```

`java.util.*`
 - Utility framework or collection framework
 - used to store collection of objects - any objects (ie) Employee, Student, String, Player, Venue

1. Collection interface - core interface in order to store collection of object
2. Collections class - provided with static methods or static algorithm that supports util package

Collection interface
 - core interface in order to store collection of object

Methods

1. boolean add(Object o) - add single object
2. boolean addAll(Collection c) - add multiple object
3. boolean remove(Object o) - remove single object
4. boolean removeAll(Collection c) - remove multiple object
5. boolean contains(Object o) - check single object present in Collection or not
6. boolean containsAll(Collection c) - check multiple object present in Collection or not
7. boolean retainAll(Collection c) - remove from the target collection all the elements that are not contained in the specified collection
8. int size() - return number of objects present in the collection
9. boolean isEmpty() - to check collection is empty or not
10. Object[] toArray() - return a array containing all elts in collection
11. String toString()
12. boolean equals(Object o)
13. Iterator iterator()
14. ListIterator listIterator()

List interface
 - Ordered and duplicate elements allowed

Methods

1. void add(int index, Object o) - add single object at particular index position
2. boolean addAll(int index, Collection c) - add multiple object at particular index position
3. Object get(int index) - return single object present at particular index position

4. int indexOf(Object o) - return position of first occurrence of object in given collection
5. int lastIndexOf(Object o) - return position of last occurrence of object in given collection
6. Object remove(int index) - remove single object present at particular index position
7. Object set(int index, Object val) - replace an object at particular index
8. List subList(int start, int end) - return part of list from start to end-1

Set interface

- unordered and no duplicate elements allowed

SortedSet interface

- sort the elts of Set interface

Methods

1. Object first() - return first elt
2. Object last() - return last elt
3. SortedSet subSet(Object from, Object end) - return part of set from start to end-1
4. SortedSet headSet(Object elt) - return all elts present before specified elt
(1 2 3 4 5).headSet(3) //1 2
5. SortedSet tailSet(Object elt) - return all elts present after specified elt
(1 2 3 4 5).tailSet(3) //4 5

List interface

- Ordered and duplicate elements allowed

int a[] = new int[3]; - Static array

1. ArrayList class

- It is similar to array but it is called as dynamic array (ie) we can increase or decrease its size at runtime

Syntax: public class ArrayList extends AbstractList implements List, Cloneable, Serializable, RandomAccess

- used for faster retrieval of data and slower in insertion and deletion
- default capacity is 10

Constructor

1. ArrayList()
2. ArrayList(int capacity)
3. ArrayList(Collection c)

```
public class Main {
    public static void main(String...args) {
        ArrayList l1=new ArrayList();
        System.out.println(l1.size()); //0
        l1.add("A");
        l1.add("B");
        l1.add(10); //Autoboxing
        System.out.println(l1.size()); //3
        //whenever we print object of any collection inside []
        System.out.println(l1); //[[A,B,10]

        ArrayList l2=new ArrayList();
        System.out.println(l2.size()); //0
        l2.addAll(l1);
        System.out.println(l2); //[[A,B,10]

        System.out.println(l1.contains(10)); //true
        System.out.println(l1.containsAll(l2)); //true
        System.out.println(l1.indexOf(10)); //2

        List l3=new ArrayList(); //DMD
        l3.add("A");
        l3.add("B");
        l3.add(1);
        l3.add(2);
        l3.add(3);
        System.out.println(l3); //[[A,B,1,2,3]

        List l4=new ArrayList();
        l4.add("A");
        l4.add("B");
    }
}
```

```

        l4.add(1);
        System.out.println(l4); // [A,B,1]

        l3.addAll(l4);
        System.out.println(l3); // [A,B,1]
    }
}

```

Generics

- Available from JDK1.5 onwards
- used to specify what type of object stored in the collection, avoid type casting
- denoted by <>

List<String> l1=new ArrayList<String>(); //JDK1.5
List<Integer> l2=new ArrayList<>(); //From JDK1.7

```

public class Main {
    public static void main(String...args) {
        List<String> l1=new ArrayList<>();
        System.out.println(l1.size()); //0
        l1.add("C");
        l1.add("F");
        l1.add(1,"O");
        l1.add("T");
        l1.add("R");
        //C,O,F,T,R
        System.out.println(l1); // [C,O,F,T,R]
        System.out.println(l1.get(2)); //F
        System.out.println(l1.contains("S")); //false
        List<String> l2=new ArrayList<>();
        l2.add("C");
        l2.add("F");
        System.out.println(l1.containsAll(l2)); //true
        System.out.println(l1.indexOf("T")); //3
        l1.remove("T"); // [C,O,F,R]
        l1.remove(1);
        System.out.println(l1); // [C,F,R]
    }
}

```

2. LinkedList class

- It is also called dynamic array
- used for faster insertion and deletion, slower in selection

Syntax: public class LinkedList extends AbstractSequentialList implements List, Cloneable, Serializable, Deque

Deque - Double Ended Queue - do insertion and deletion at both ends

Constructors

1. LinkedList()
2. LinkedList(Collection c)

Methods

1. void addFirst()
2. void addLast()
3. Object getFirst()
4. Object getLast()
5. Object removeFirst()
6. Object removeLast()
7. void push(Object o)
8. Object pop()
9. boolean offerFirst(Object o) - insert specified elt front of the list
10. boolean offerLast(Object o) - insert specified elt last of the list
11. Object peekFirst() - retrieves first elt of the list but does not remove
12. Object peekLast() - retrieves last elt of the list but does not remove
13. Object pollFirst() - retrieves first elt of the list and remove
14. Object pollLast() - retrieves last elt of the list and remove

```

public class Main {
    public static void main(String...args) {

```

```

LinkedList<Integer> l1=new LinkedList<>();
l1.add(3);
l1.add(5);
l1.add(2);
l1.add(1,7);
l1.addFirst(8);
l1.addLast(9);
System.out.println(l1.size()); //6
System.out.println(l1); //[8,3,7,5,2,9]
System.out.println(l1.get(5)); //9
System.out.println(l1.getFirst()); //8
System.out.println(l1.getLast()); //9
l1.remove(1); //8,7,5,2,9
l1.removeFirst(); //7,5,2,9
l1.removeLast();
System.out.println(l1); //[7,5,2]
System.out.println(l1.peekFirst()); //7
System.out.println(l1.pollLast()); //2
System.out.println(l1); //[7,5]
}
}

```

3. Vector class

- It is a legacy(older) class
- It is also called dynamic array, but it is synchronized or threadsafe
- Default capacity is 10

Syntax: public class Vector extends AbstractList implements List,Cloneable, Serializable

Constructor

1. Vector()
2. Vector(int capacity)
3. Vector(int capacity,int increment)
4. Vector(Collection c)

```

public class Main {
    public static void main(String...args) {
        Vector<Integer> v=new Vector<>(3,2);
        System.out.println(v.size()); //0
        System.out.println(v.capacity()); //3
        v.add(1);
        v.add(2);
        v.add(3);
        v.add(4);
        System.out.println(v.size()); //4
        System.out.println(v.capacity()); //5
        v.add(5);
        v.add(6); //5+2
        System.out.println(v.size()); //6
        System.out.println(v.capacity()); //7
        System.out.println(v); //[[1,2,3,4,5,6]
    }
}

```

Set interface

- unordered and no duplicates allowed

1. HashSet class

- does not maintain any insertion order and the elts are inserted in the basis of their hashCode value
- default capacity is 16

Syntax: public class HashSet extends AbstractSet implements Set, Cloneable, Serializable

Constructor

1. HashSet()
2. HashSet(int capacity)
3. HashSet(int capacity,float fillratio)

fillratio ranges from 0.0 to 1.0 - default is 0.75
4. HashSet(Collection c)

```
Set<Integer> hs=new HashSet<>(); Map<Integer, Object> hm=new HashMap<>();
```

```
sop(hs.size()); //0
hs.add(2); hm.put(2,PRESENT);
hs.add(3); hm.put(3,PRESENT);
hs.add(2); hm.put(2,PRESENT);
sop(hs.size()); //2
sop(hs); // [2,3]
```

When we create HashSet, it internally creates HashMap(unique key value pair), with key as what elt we are inserting into the hashset and value as dummy object called PRESENT

If hm.put(key,value) returns null then the statement hm.put(2,PRESENT)==null will return true then the element will be added into hashset

If hm.put(key,value) returns old value of key then the statement hm.put(2,PRESENT)==null will return false then the element will not be added into hashset

```
Set<Integer> hs=new HashSet<>(3);
hs.add(1);
hs.add(2);
hs.add(3);
hs.add(4); 3*0.75=2.25+3=5.25
hs.add(5);
hs.add(6); 5.25*0.75=3+5.25=8.25
```

2. LinkedHashSet class

- used to maintain the insertion order of the set

Syntax: public class LinkedHashSet extends HashSet implements Set,Cloneable, Serializable

Constructor

1. LinkedHashSet()
2. LinkedHashSet(int capacity)
3. LinkedHashSet(int capacity,float fillratio)
fillratio ranges from 0.0 to 1.0 - default is 0.75
4. LinkedHashSet(Collection c)

3. TreeSet class

- Print the elements of Set interface in Sorted format(ie)numbers in ascending, string in alphabetical order

Syntax: public class TreeSet extends AbstractSet implements NavigableSet, Cloneable, Serializable

Constructor

1. TreeSet()
2. TreeSet(Comparator c)
3. TreeSet(Collection s)
4. TreeSet(SortedSet s)

```
public class Main {
    public static void main(String...args) {
        Set<Integer> hs=new HashSet<>();
        hs.add(20);
        hs.add(3);
        hs.add(11);
        hs.add(4);
        hs.add(1);
        System.out.println(hs); //random order

        LinkedHashSet<Integer> ls=new LinkedHashSet<>();
        ls.add(20);
        ls.add(3);
        ls.add(11);
        ls.add(4);
        ls.add(1);
        System.out.println(ls); // [20,3,11,4,1]

        TreeSet<Integer> ts=new TreeSet<>();
        ts.add(20);
        ts.add(3);
        ts.add(11);
        ts.add(4);
        ts.add(1);
        System.out.println(ts); // [1,3,4,11,20]
```

```

TreeSet ts1=(TreeSet)ts.descendingSet();
System.out.println(ts1); // [20,11,4,3,1]
}
}

```

To print individual elements of collection - 5 ways

1. Normal for loop
2. for each stmt
3. Using Iterator interface
 - used to print individual elements of collection only in forward direction
 - Using Iterator iterator(), we can create object for Iterator
 - Methods
 1. boolean hasNext() - check whether it contains next next elt
 2. Object next() - retrieve each elt
 3. void remove() - at time of accessing even we can remove the elt
4. Using ListIterator interface
 - used to print individual elements of collection both in forward and backward direction
 - Using ListIterator listIterator(), we can create object for ListIterator
 - Methods
 1. boolean hasNext() - check whether it contains next next elt
 2. Object next() - retrieve each elt
 3. void remove() - at time of accessing even we can remove the elt
 4. boolean hasPrevious()
 5. Object previous()
 6. void set(Object o)
 7. void add(Object o)
5. Using Enumeration interface
 - It is legacy interface
 - used to print individual elements of collection only in forward direction
 - Using Enumeration elements(), we can create object for Enumeration
 - Methods
 1. boolean hasMoreElements() - check whether it contains next next elt
 2. Object nextElement() - retrieve each elt

```

public class Main {
    public static void main(String...args) {
        List<Integer> l1=new ArrayList<>();
        l1.add(30);
        l1.add(20);
        l1.add(10);
        l1.add(24);
        System.out.println(l1); // [30,20,10,24]
    }
}

```

```

// 1. Normal FOR loop
for(int i=0;i<l1.size();i++) {
    System.out.println(l1.get(i));
}

```

```

System.out.println("Using For each stmt");
for(Integer i:l1)
    System.out.println(i);

```

```

System.out.println("Using Iterator");
Iterator<Integer> i=l1.iterator();
while(i.hasNext()) {
    // Integer i1=(Integer)i.next();
    Integer i1=i.next();
    System.out.println(i1);
}

```

```

List<Student> l2=new ArrayList<>();
l2.add(new Student(1,"Ram",23));
l2.add(new Student(2,"Sam",21));
l2.add(new Student(3,"Tam",20));
l2.add(new Student(4,"Jam",24));

```

```

ListIterator<Student> li=l2.listIterator();
while(li.hasNext()) {
    Student s1=li.next();
    System.out.println(s1.getName());
}

```

From: Senthil Kumar T
Sent: Monday, September 11, 2023 3:18 PM
To: Yerramsetti V Subrahmanyamaisatyanarayana <yerramsetti.v.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetricalmurugan Selvaraj' <vetricalmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkatesh@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tunigena Vishnu Vardhan Rao <tunigena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan.s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarrajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra.s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajednra <aherrutuja.rajednra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteeek <akasam.karteeek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjevi Chandran <siranjevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami

<sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvia Daniel Antohe Multescu <silviudaniel.antoh@hcl.com>; Kohale Siddhesh Prashant' <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzserra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheikh <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; 'Sebastian Nae' <sebastian.nae@hcl.com>; Sayojo P <sayojo.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarasarwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhone <sakshi.dhone@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujavarapu <priyanka.jujavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r_priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathiyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khatti <poonam.khatti@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vitthal Benki <nagarajvitthal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam_jyostna@hcl.com>; Aravindh L <aravindh_l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; Archana Thineta' <archana.thineta@hcl.com>; Archana C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <monisha.i@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhy@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.mohamed@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyalu <maniramkumar.siriylu@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain

<asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапала <ashini.rathnапала@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwary@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shankar@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha_s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakan.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; kalpana.golla@hcl.com; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunk@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <atinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadarai S <jabadarai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitaja@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammanna <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 7 Notes

Classification: **Confidential**

Handson

1. Exception 2(ArrayIndexOutOfBoundsException And NegativeArraySizeException)

Write a program to get the number of overs and the runs scored in each over. Get the over number from the user and display number of runs scored in that over. Let

- number of overs be the array size
- over number be the index of the array+1
- runs be the array elements

This program may generate

1. NegativeArraySize Exception when the number of overs is negative
2. ArrayIndexOutOfBoundsException when the over number that is searched is beyond the specified over numbers.

Use exception handling mechanisms to handle these exceptions. Use a single catch block. In the catch block, print the class name of the exception thrown.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input/Output 1:

Enter the number of overs

3

Enter the number of runs for each over

8

15

12

Enter the over number

2

Runs scored in this over : 15

Sample Input/Output 2:

Enter the number of overs

3

Enter the number of runs for each over

8

15

12

Enter the over number

4

java.lang.ArrayIndexOutOfBoundsException

Sample Input/Output 3:

Enter the number of overs

-1

Enter the number of runs for each over

java.lang.NegativeArraySizeException

2. Custom Exceptions [Age]

Write a program to get the name and age of the player from the user and display it.

player name is a string

player age is an integer value

Note : The player is eligible to participate in IPL when their age is 19 and above

This program may generate

1. InvalidAgeRange Custom Exception when the player's age is below 19

Use exception handling mechanisms to handle these exceptions

Create a class called CustomException which extends Exception and it includes constructor to initialize the message.

Use appropriate exception handling mechanisms to handle these exceptions

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input/Output 1:

Enter the player name

Albie Morkel

Enter the player age

35

Player name : Albie Morkel

Player age : 35

Sample Input/Output 2:

Enter the player name

Ishan Kishan

Enter the player age

16

CustomException: InvalidAgeRangeException

3. TeamNameNotFoundException

Write a program to get the two team names i.e expected Runner and Winner team of IPL season 4 and display it.

Team name is a string

Note : The team name given below are only eligible to take part in IPL season 4

Chennai Super Kings

Deccan Chargers

Delhi Daredevils

Kings XI Punjab

Kolkata Knight Riders

Mumbai Indians

Rajasthan Royals

Royal Challengers Bangalore

This program may generate TeamNameNotFoundException Custom Exception when the expected team entered is not present in the above eligible teams list for IPL season 4.

Use exception handling mechanisms to handle these exceptions

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output 1:

Enter the expected winner team of IPL Season 4

Chennai Super Kings

Enter the expected runner Team of IPL Season 4

Mumbai Indians

Expected IPL Season 4 winner: Chennai Super Kings

Expected IPL Season 4 runner: Mumbai Indians

Sample Input and Output 1:

Enter the expected winner team of IPL Season 4

Pune Warriors

TeamNameNotFoundException: Entered team is not a part of IPL Season 4

java.lang.*

1. Object

2. Wrapper class

3. String

4. StringBuffer

5. StringBuilder

Java (2 concept)

Error - generated at compile time - syntactical/human made error

Exception - generated at runtime - logical error, eg: Divide by 0, file not found

6. Throwable class

- used to handle exception handling in Java

2 types of Exception

1. Checked Exception

- all subclasses of Exception class excluding RuntimeException. Even though we have written 100% correct prg also, ur code has to be surrounded by try/catch block or throws keyword otherwise ur prg will not compile

- Will insist the programmer to surround the code using try/catch block or throws keyword otherwise ur prg will not compile

2. Unchecked Exception

- all subclasses of RuntimeException class

- Even though ur prg contains an exception, the compiler will just compile the program but at runtime we get related exception

- Will not insist the programmer to surround the code using try/catch block or throws keyword, the prg will compile but at runtime we get related exception

Constructors

1. Throwable()

2. Throwable(String msg)

3. Throwable(String msg, Throwable t)

Types of Exception

1. ArithmeticException - divide anything by 0

2. `ArrayIndexOutOfBoundsException` - only for datatype array

```
int a[] = new int[3];
a[4] = 1;
```

3. `StringIndexOutOfBoundsException` - only for string array

```
String a[] = new String[3];
a[4] = "1";
```

4. `NegativeArraySizeException`

```
int a[] = new int[-3];
```

5. `NumberFormatException`

```
int a = Integer.parseInt("abc");
```

6. `ArrayStoreException`

```
int a[] = new int[3];
```

```
a[0] = "one";
```

7. `NullPointerException`

```
class A{
```

```
    void show(){
```

```
}
```

```
}
```

class Main {PSVM {

```
    A a = new A();
```

```
    a.show();
```

```
    a = null;
```

```
    a.show();
```

8. `ClassCastException`

```
class A{
```

```
}
```

class B extends A {}

A a = new A();

B b = new B();

a = b; //correct

b = a; //Exception

5 keywords

1. `try`

- Program to be monitored for exception has to be put inside try block

2. `catch`

- used to catch the exception generated, mainly used to print user defined messages when an exception occurs

3. `finally`

- optional stmt, it will be executed everytime irrespective of exception occurs or not

- used for closing the resource or cleaning the memory

- used in 3 cases - file, database, socket programming

```
try {
fp = fopen("a.txt", "r");
read operation
}
catch(Exception e) {
}
finally {
fclose(fp);
}
```

3 ways

1. `try{`

2. `try{`

3. `try{`

```
}
```

```
}
```

```
}
```

catch(Exception e){

catch(Exception e){

finally {

`finally{`

}

- We should not write anything between trycatch and finally

To get input from user

1. Using Scanner/BufferedReader

2. Directly define in prg itself

3. Using CommandLine argument

- We give input while running the prg in command line

- When we use args argument inside the prg, then for that prg we have to give input through command line

- Each command line should be separated by space

```
public class Main {
```

```
public static void main(String[] args) {
    for(int i=0;i<args.length;i++) {
        System.out.println(args[i]);
    }
}
```

Rigth Click - Run as - Run configuration - Goto Arguments tab - Under Program Arguments we have to give input separated with space

```
public class Main {
    public static void main(String[] args) {
        try {
            int a=Integer.parseInt(args[0]);
            int c=10/a;
            System.out.println(c);
        }
        catch(ArithmaticException e) {
            System.out.println("Divide by 0 "+e);
        }
    }
}
```

Multi catch statement

- single try can contain multiple catch blocks

```
public class Main {
    public static void main(String[] args) {
        try {
            int a=Integer.parseInt(args[0]);
            int c=10/a;
            System.out.println(c);
            int b[]={23};
            b[50]=100;
        }
        catch(ArithmaticException e) {
            System.out.println("Divide by 0 "+e);
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Array Index: "+e);
        }
    }
}
```

- Whenever we define general class called Exception or Throwable, it should be always present in last catch block otherwise it leads to compilation error

```
public class Main {
    public static void main(String[] args) {
        try {
            int a=Integer.parseInt(args[0]);
            int c=10/a;
            System.out.println(c);
            int b[]={23};
            b[50]=100;
        }
        /*catch(/*Exception e*/ Throwable e) {
            System.out.println(e);
        }*/
        //Compilation error
        catch(ArithmaticException e) {
            System.out.println("Divide by 0 "+e);
        }
        catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("Array Index: "+e);
        }
        catch(NumberFormatException e) {
            System.out.println(e);
        }
        catch(NullPointerException e) {
            System.out.println(e);
        }
        catch(/*Exception e*/ Throwable e) {
```

```
        System.out.println(e);
    }
```

```
}
```

- From JDK1.7 version, we define multiple exception in a single catch block using |

```
public class Main {
    public static void main(String[] args) {
        try {
            int a=Integer.parseInt(args[0]);
            int c=10/a;
            System.out.println(c);
            int b[]={23};
            b[50]=100;
        }

        catch(ArithmaticException|ArrayIndexOutOfBoundsException
              |NumberFormatException|NullPointerException e) {
            System.out.println(e);
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

4. throw keyword

- used to manually throw an exception
- whenever it invokes throw keyword, it will automatically goes to related catch block

Syntax: throw new Exception(String msg);

```
public class Main {
    static void demo() {
        try {
            throw new NullPointerException("Hello");
        }
        catch(NullPointerException e) {
            System.out.println("Caught");
            throw e;
        }
    }
    public static void main(String[] args) {
        try {
            demo();
        }
        catch(NullPointerException e) {
            System.out.println("Recaught");
        }
    }
}
```

5. throws keyword

- throws keyword is used to declare an exception, and used only in methods
- Used to indicate that the method might throw one of the exception

```
public class Main {
    static void demo() throws NullPointerException {
        throw new NullPointerException("Hello");
    }
    public static void main(String[] args) {
        try {
            demo();
        }
        catch(NullPointerException e) {
            System.out.println("Recaught");
        }
    }
}
```

```

    }
}

public class Main {
    static void demoA() {
        try {
            System.out.println("Inside demoA");
            throw new RuntimeException();
        }
        /*catch(RuntimeException e) {
            System.out.println("Exception caught inside demoA");
        }*/
        finally {
            System.out.println("demoA finally");
        }
    }
    static void demoB() {
        try {
            System.out.println("Inside demoB");
            return;
        }
        finally {
            System.out.println("demoB Finally");
        }
    }
    public static void main(String[] args) {
        try {
            demoA();
        }
        catch(RuntimeException e) {
            System.out.println("Exception caught");
        }
        demoB();
    }
}

```

User defined Exceptions

- Your userdefined exception class should extend Exception class and override toString()

```

class A extends Exception {
}

class NotValidAgeException extends Exception{
    String s1="";
    public NotValidAgeException(String s1) {
        this.s1=s1;
    }
    @Override
    public String toString() {
        return s1;
    }
}
public class Main {
    static void validateAge() throws NotValidAgeException {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter age");
        int age=sc.nextInt();
        if(age<18) {
            throw new NotValidAgeException("Your age is not eligible");
        }
        else {
            System.out.println("Your age is eligible");
        }
    }
    public static void main(String[] args) {
        try {
            validateAge();
        }

```

```

        }
        catch(NotValidAgeException e) {
            System.out.println(e);
        }
    }
}

```

Assertions

- Available from JDK1.5 onwards
 - used to check boolean conditions at runtime
- Syntax: assert <>expression>;
 assert <>expression>:String message;
 - By default assertion is disabled in Java, while running we have to enable the assertion using -ea options
 - Provided with assert keyword, AssertionError class is unchecked exception because it is inherited from Error class
 - If assert condition fails then it will throw AssertionError class

```

public class Main {
    static double withdraw(double balance, double amount) {
        assert (balance>=amount); //Java will just ignore this line
        return (balance-amount);
    }
    public static void main(String[] args) {
        System.out.println(withdraw(1000,500)); //500.0
        System.out.println(withdraw(1000,2000)); //-1000.0
    }
}

```

```

public class Main {
    static double withdraw(double balance, double amount) {
        assert (balance>=amount);
        return (balance-amount);
    }
    public static void main(String[] args) {
        System.out.println(withdraw(1000,500)); //500.0
        System.out.println(withdraw(1000,2000)); //throws AssertionError class
    }
}

```

```

public class Main {
    static double withdraw(double balance, double amount) {
        assert (balance>=amount):"Balance is insufficient";
        return (balance-amount);
    }
    public static void main(String[] args) {
        System.out.println(withdraw(1000,500)); //500.0
        System.out.println(withdraw(1000,2000)); //-1000.0
    }
}

```

try with resources

- Available from JDK1.7 onwards
- Used to close the resources automatically at the end of stmt, using AutoCloseable interface

Syntax:
 try(resources) {
 }
 catch(Exception e){
 }

```

public class Main {
    public static void main(String[] args) {
        String line="";
        try(BufferedReader br=new BufferedReader(new FileReader("C:\\Spring\\JavaExample\\src\\com\\pack\\Main.java")));
            PrintWriter pw=new PrintWriter(new File("a.txt"));
            while((line=br.readLine())!=null) {
                System.out.println(line);
            }
        }
}

```

```

        catch(Exception e) {
            System.out.println(e);
        }
    }
}

```

From JDK1.9 onwards, enhancement in try with resources, we can define the resources outside the try block and we can use it

```

public class Main {
    public static void main(String[] args) throws Exception {
        String line="";
        BufferedReader br=new BufferedReader(new FileReader("C:\\Spring\\JavaExample\\src\\com\\pack\\Main.java"));
        PrintWriter pw=new PrintWriter(new File("a.txt"));
        try(br;pw){
            while((line=br.readLine())!=null) {
                System.out.println(line);
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}

```

Exception handling in method overriding

1. If superclass method does not declare an exception , then subclass overridden method cannot declare the checked exception but it can declare unchecked exception

```

class A {
    void add() {
    }
}
class B extends A {
    void add() throws ArithmeticException { //correct
    }
}

class A {
    void add() {
    }
}
class B extends A {
    void add() throws IOException { //error
    }
}

```

2. If superclass method declares an exception , then subclass overridden method can declare the same exception,subclass exception or no exception but it cannot declare parent exception

```

class A {
    void add() throws Exception {
    }
}
class B extends A {
    void add() throws Exception { //correct
    }
}

class A {
    void add() throws Exception {
    }
}
class B extends A {
    void add() { //correct
    }
}

```

```

    }
}

class A {
    void add() throws ArithmeticException {

    }
}
class B extends A {
    void add() throws Exception { //error

    }
}

class A {
    void add() throws Exception {

    }
}
class B extends A {
    void add() throws ArithmeticException { //correct

    }
}

```

From: Senthil Kumar T

Sent: Friday, September 8, 2023 3:28 PM

To: Yerramsetti V Subrahmanyamaisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetricalmurugan Selvaraj' <vetricalmurug.selva@hcl.com>; Venugopala Reddy Kopula <venugopalared.koppu@hcl.com>; Venkatesh S <venkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni_sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarakumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajendran <aherrutuja.rajendran@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; 'sridhar.at@hcl.com' <sridhar.at@hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.anton@hcl.com>; Kohale Siddhesh Prashant <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmane <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyK@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra

<alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar_srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae' <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_si@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupeesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; 'ricardo.lopezporras@hcl.com' <ricardo.lopezporras@hcl.com>; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; 'rahulvenkat.gone@hcl.com' <rahulvenkat.gone@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujuvarapu <priyanka.jujuvarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pwankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premisingh <padya.premisingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam_jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadrapi <metkarisaf.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydi@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashisth@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Athukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапala <[ashini.rathnапала@hcl.com](mailto:ashini.rathnапala@hcl.com)>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshitji.somes@hcl.com](mailto:bhosalekshitji.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritisak@hcl.com>; Aswitha J <aswitha.j@hcl.com>;

Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondou Srinidhi <kondou.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankar Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha-s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumarrt@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <katmatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; 'kalpana.golla@hcl.com' <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; JudsonSam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Halley Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhima@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barapatla Venkatesh <barapatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saita@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Varas Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Dontu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaanilkumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dash@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommura@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saita@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 6 Notes

Classification: **Confidential**

Handson

1. IPL Merchandise is an alternative form of revenue and brings substantial income.

Another merchandise company called "Fundas" has come up with a native flair offering customized mugs, team badges, helmets, etc. The company has decided to print a special code on the merchandises. The special code contains the captions of the IPL teams along with the jersey number of the celebrity players of the corresponding team on the merchandises. The code to be printed is considered valid for printing only if their:

--> first word is one of the team's caption [RCB, MI, CSK, SRH, KXIP, DD, KKR, RPSG and GL]

--> second word is the jersey number

Write a program that reads a string S corresponding to the team name and jersey number and validates if the name is eligible to be printed on the stocks.

Include a class UserMainCode with a static method called validateTeam which accepts a string and its return type is bool. In this method display the details as given in sample input and output.

Create a Class Main which would be used to accept two Strings and call the static method called validateTeam present in UserMainCode.

Input Format:

First line of the input is a string S, that corresponds to the team name and jersey number.

Output Format:

Output should display "Valid" if the team name and jersey number is valid for printing. Otherwise print "Invalid".

Sample Input 1 :

CSK 7

Sample Output 1 :

Valid

Sample Input 2 :

RCB1 18

Sample Output 2 :

Invalid

2. In Cricket it is so common that the players are often called by their last names. Few cricketers have their origins from the same regions of the country which is why they have common last names. Sandeep is now entrusted with yet another task that is to check for the last names of players.

Given two player's names P1 and P2, Sandeep has to write a program to find if the last name of the two given players are the same. Help him do this using String Builder method.

Include a class UserMainCode with a static method called display which accepts two strings and its return type is void. In this method display the details as given in sample input and output.

Create a Class Main which would be used to accept two Strings and call the static method called display present in UserMainCode.

Input Format:

First line of the input is a string P1, that corresponds to the first player's name.

Second line of the input is a string P2, that corresponds to the second player's name.

Output Format:

Output should print in a single line "Yes" if the last names of the players are the same. Otherwise print "No".

Sample input

Mohit Sharma

Rohit Sharma

Sample output

Yes

Sample input

Mohit Sharma

Virat Kohli

Sample output

No

3. Write a program to read a string and find whether it is a valid string or not.

Validation Rules :

1. It accepts the player name as a string and is appended with the player's skill.

2. It ends with a symbol ,the symbol is '*' if the player is a raider and it is '#' if the player is a defender.

3. Player's first name,last name and skill should start with upper case letters

Include a class UserMainCode with a static method validatePlayer which accepts a string. In this method check whether the given player name is valid as per the validation rules mentioned above. The return type is Boolean.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of a string "Valid" or "Invalid".

Refer sample output for formatting specifications.

Sample Input 1:

Anup Kumar-Raider*

Sample Output 1:

Valid

Sample Input 2:

Surender Nada-Defender#

Sample Output 2:

Valid

Sample Input 3:

Rohit Rana-Defender*

Sample Output 3:

Invalid

4. Write a program to read a string and find whether it is a valid string or not.

Validation Rules :

1. It accepts the player name as a string ,player name length should be between 4 to 20.

2..It starts with four digit number representing the player's debutant year.

3.It ends with single digit number representing the number of IPL season played by the player.

Include a class UserMainCode with a static method validatePlayer which accepts a string. In this method check whether the given player name is valid as per the validation rules mentioned above. The return type is Boolean.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of a string "Valid" or "Invalid".

Refer sample output for formatting specifications.

Sample Input 1:

2013Mohit Sharma3

Sample Output 1:

Valid

Sample Input 2:

200Mohammad Kaif

Sample Output 2:

Invalid

5. T20 IPL board get the number of player names and finds the players having the string "Sharma" in their name. Write a program to get the player names in an array and use Contains method to get the players whose name has the string "Sharma".

Input and Output Format:

Refer sample input and output for formatting specifications.

Sample Input/Output 1:

Enter number of players

5

Enter player names

Rohit Sharma

Adam Smith

Ishant Sharma

Mohit Sharma

Jaspirit Bumrah

Rohit Sharma

Ishant Sharma

Mohit Sharma

java.lang.* package

1. Object class

2. Wrapper class

- Float class

- Double class

-Integer,Short,Byte,Long wrapper class

Constructors

```

1. Integer(int i)
    Integer i=new Integer(20);
Integer(String s) throws NFE
    Integer i=new Integer("20");
    Integer i1=new Integer("abc");

2. Short(short s)
    Short s1=new Short(23); //error
    Short s2=new Short((short)23); //correct
Short(String s) throws NFE
    Short s3=new Short("23");

3. Byte(byte b)
    Byte b1=new Byte(12); //error
    Byte b2=new Byte((byte)12);
Byte(String s) throws NFE
    Byte b3=new Byte("12");

4. Long(long l)
    Long l1=new Long(14); //correct
    Long l1=new Long(14l); //correct
Long(String s) throws NFE
    Long l1=new Long("14");

```

Methods

1. static int parseInt(String s)
2. static int parseInt(String s,int radix)

```

public class Main {
    public static void main(String[] args) {
        String s="42";
        int i=Integer.parseInt(s);
        System.out.println(i); //42
        int j=Integer.parseInt(s,5);
        //2^5^0=2
        //4^5^1=20
        System.out.println(j); //2
        System.out.println(Integer.toBinaryString(2)); //10
        System.out.println(Integer.toOctalString(9)); //11
        System.out.println(Integer.toHexString(10)); //a
        System.out.println(Integer.MAX_VALUE);
        System.out.println(Integer.MIN_VALUE);
        System.out.println(Integer.TYPE); //int
    }
}

```

5. Boolean class
 - used to perform operation on boolean datatype

Constructor

```

1. Boolean(boolean b)
Boolean b1=new Boolean(true);
sop(b1); //true
Boolean b2=new Boolean(false);
sop(b2); //false
Boolean b3=new Boolean(True);
sop(b3); //error
Boolean b4=new Boolean(Helo);
sop(b4); //error

2. Boolean(String s) - we can give only true in any format it will return true, otherwise it returns false
Boolean b1=new Boolean("true");
sop(b1); //true
Boolean b2=new Boolean("false");
sop(b2); //false
Boolean b3=new Boolean("True");
sop(b3); //true
Boolean b4=new Boolean("hello");
sop(b4); //false
Boolean b5=new Boolean("TRUE");
sop(b5); //true

```

6. Character class

- used to perform operation on char datatype

Constructor

- Character(char c)

Methods

- static boolean isDigit(char c)
- static boolean isLetter(char c)
- static boolean isLetterOrDigit(char c)
- static boolean isUpperCase(char c)
- static boolean isLowerCase(char c)
- static boolean isSpace(char c)
- static char toUpperCase(char c)
- static char toLowerCase(char c)

Autoboxing and Unboxing

- Introduced from JDK1.5 onwards
- Autoboxing is automatic conversion of primitive datatype to wrapper class
- Unboxing is automatic conversion of wrapper class to datatype

With autoboxing Without autoboxing

```
Integer i;                              Integer i;
int j;                                 int j;
i=25; //Autoboxing                    i=new Integer(25);
j=i; //Unboxing                      j=i.intValue();
```

3. String class

- String is final class in java.lang.*
- Fixed length of char, hence it is called immutable class (ie) we cant increase or decrease its size at runtime

Syntax: public class String extends Object implements Serializable, Cloneable, Comparable, CharSequence

Constructor

- String()
- String(String s)
- String(byte[] b)
- String(byte[] b, int start,int numofbytes)
- String(char[] c)
- String(char[] c, int start,int numofchars)
- String(StringBuffer s)

Methods

- String toString() - String representation of an object
- char charAt(int pos) - return a single char
- void getChars(int start,int end,char[] buf, int targetstart) - return a group of char
- byte[] getBytes() - convert string to byte[]
- char[] toCharArray() - convert String to char[]
- boolean equals(String s) - check equality of content by taking case into consideration
- boolean equalsIgnoreCase(String s) - check equality of content by without taking case into consideration
- ===(equal versus) - check to equality of object reference
- boolean startsWith(String s)
- boolean endsWith(String s)
- String toLowerCase()
- String toUpperCase()
- int compareTo(String s) - compare 2 strings taking case into consideration and do sorting, if two values are equal it returns 0, greater means +1, lesser means -1
- int compareIgnoreCase(String s) - compare 2 strings without taking case into consideration and do sorting, if two values are equal it returns 0, greater means +1, lesser means -1
- String intern() - it stores string object info in string pool

String pool is a storage space in the Java heap memory where string literals are stored. It is also known as String Constant Pool or String Intern Pool. It is privately maintained by the Java String class. By default, the String pool is empty.

In stack memory, only the primitive data types like- int, char, byte, short, boolean, long, float and double are stored. Whereas, in the heap memory, non-primitive data types like strings are stored. A reference to this location is held by the stack memory.

The String Pool is empty by default, and it is maintained privately by the String class.

When we create a string literal, the JVM first checks that literal in the String Constant Pool. If the literal is already present in the pool, its reference is stored in the variable.

However, if the string literal is not found, the JVM creates a new string object in the String Constant Pool and returns its reference.

```
String s1="Hello"; //literal - string pool
String s2="Hello";
String s3="hi";

String s4=new String("Gello"); - stored in heap memory
String s5=new String("Gello"); - created new memory in heap
s4.intern(); s4 into string pool
```

16. int indexOf(char c)
 int indexOf(String s)
 int indexOf(int charval,int startindex)
 int indexOf(String s,int index)
 - used to return the position of first occurrence of char in given string, if not found it returns -1

17. int lastIndexOf(char c)
 int lastIndexOf(String s)
 int lastIndexOf(String s,int index)
 - used to return the position of last occurrence of char in given string, if not found it returns -1

18. String substring(int start) - return part of string from start position

19. String substring(int start,int end) - return part of string from start to end-1

20. int length()

21. String concat(String s) or + operator

22. String trim() - remove leading and trailing space

23. String replace(char old,char new)

24. String[] split(String delimiter)

String[] split(String delimiter,int limit)
 - used to split the string based on delimiter

25. boolean matches(String regex) - match the string based on regex

26. boolean regionMatches(boolean ignorecase,int toffset,String search,int ooffset,int length) - used to match part of a string with another string

27. int codePointAt(int index)

28. int codePointBefore(int index)

29. int codePointCount(int beginindex, int endindex)

30. boolean contains(String s)

31. static String join(String delimiter, String...s) - used to join the string based on delimiter

32. static String copyValueOf(char[] c)

static String copyValueOf(char[] c,int offset,int count)

33. static String format(String format, Object val)

```
int a=13;
printf("%d",a);
```

%+-Owspecifier.precision

+ - add space before number

-- add space after number

0 - pad with zero

w - total width

specifier - d,f,x,o,s

precision - after decimal place

```
public class Main {
    public static void main(String[] args) {
        byte[] b= {65,66,67,68,69};
        String s1=new String(b);
        System.out.println(s1); //ABCDE
        String s2=new String(b,1,3);
        System.out.println(s2); //BCD

        char c[]={ 'J','a','v','a'};
        String s3=new String(c);
        System.out.println(s3); //Java
        String s4=new String(c,0,2);
        System.out.println(s4); //Ja
        String s5=new String(s4);
        System.out.println(s5); //Ja

        String s6="Hello"; //literal
        System.out.println(s6.charAt(1)); //e

        String s7="This is a demo of getchars method";
    }
}
```

```
int start=10, end=14;
char buf[]=new char[end-start]; //4
s7.getChars(start, end, buf, 0);
System.out.println(buf); //demo

String s8="ABCD";
byte b1[]={s8.getBytes()};
for(byte b2:b1)
    System.out.println(b2); //65 66 67 68

char c1[]={s8.toCharArray()};
for(char c2:c1)
    System.out.println(c2); //A B C D

String s9=new String("Hello");
String s10=new String("Hello");
String s11=new String("HELLO");
System.out.println(s9.equals(s10)); //true
System.out.println(s9.equals(s11)); //false
System.out.println(s9.equalsIgnoreCase(s11)); //true
System.out.println(s9==s10); //false
String s12=s10;
System.out.println(s10==s12); //true

String s13="Hello";
String s14="Hello";
System.out.println(s13==s14); //true

System.out.println("Foobar".startsWith("Foo")); //true
System.out.println("Foobar".endsWith("bar")); //true
System.out.println("Foobar".toLowerCase()); //foobar
System.out.println("Foobar".toUpperCase()); //FOOBAR
System.out.println("hello".compareTo("hello")); //0
System.out.println("cat".compareTo("hello")); //-5
System.out.println("hello".compareTo("cat")); //5
System.out.println("hell".compareTo("heck")); //9

String s15="It is the time for all good men to come to their "
+ "country and pay their due tax";
System.out.println(s15.indexOf('t')); //1
System.out.println(s15.indexOf("the")); //6
System.out.println(s15.indexOf("THE")); //1
System.out.println(s15.indexOf(" ")); //2
System.out.println(s15.indexOf(116,7)); //10
System.out.println(s15.indexOf("the",10)); //43
System.out.println(s15.lastIndexOf('t')); //75
System.out.println(s15.lastIndexOf("the")); //65
System.out.println(s15.lastIndexOf("the",65)); //43

String s16="Helloworld";
System.out.println(s16.substring(6)); //orld
System.out.println(s16.substring(2,6)); //llow - start to end-1

String s17="Hello";
System.out.println(s17.length()); //5
System.out.println(s17.concat("world")); //Helloworld
System.out.println(s17.length()); //5
String s18=s17.concat("world");
System.out.println(s17); //Hello
System.out.println(s18); //Helloworld

String s19=" Hello world ";
System.out.println(s19.length()); //13
System.out.println(s19.trim()); //Hello world
System.out.println(s19.length()); //13
String s20=s19.trim();
System.out.println(s20.length()); //11

System.out.println("Hello".replace('e', 'i')); //Hillo

String s21="one-two-three";
```

```

String t1[]=s21.split("-");
for(String t:t1)
    System.out.println(t); //one two three

String s22="one.two.three";
String t2[]=s22.split([file://]\.\.));
for(String t:t2)
    System.out.println(t); //one two three

String s23="A*bunch*of*stars";
String t3[]=s23.split("\\*");
for(String t:t3)
    System.out.println(t); //A bunch of stars

String s24="A*bunch*of*stars";
String t4[]=s24.split("\\*",3);
for(String t:t4)
    System.out.println(t); //A bunch of*stars

String s25="string is a class";
String t5[]=s25.split("s");
for(String t:t5)
    System.out.println(t); //tring i
    // a cla

String s26="No concession, no concillation, no compromise and just give and take policy";
String t6[]=s26.split("concession|concillation|compromise|(give and take)");
for(String t:t6)
    System.out.println(t); //No
    //, no
    //, no
    //and just
    //policy

String s27="Welcome to Java";
System.out.println(s27.matches("(.* to PHP"))); //false
System.out.println(s27.matches("Welcome to (.*)")); //true
System.out.println(s27.matches("Java")); //false
System.out.println(s27.matches("(.* to (.*)")); //true

String s28="ABC Windows test"; //Windows test
System.out.println(s28.regionMatches(4, "windows", 0, 7)); //false
System.out.println(s28.regionMatches(true,4, "windows", 0, 7)); //true

System.out.println("abcd".codePointAt(0)); //97
System.out.println("abcd".codePointBefore(2)); //98
System.out.println("abcdefg".codePointCount(0, 5)); //5
// System.out.println("abcdefg".codePointCount(0, 9)); //exception
System.out.println("Hello".contains("e")); //true
System.out.println("Hello".contains("E")); //false

System.out.println(String.join("-", "one", "two", "three")); //one-two-three

char c2[]={'a','b','c','d','e','f','g'};
String s29="";
System.out.println(s29.copyValueOf(c2)); //abcdefg
System.out.println(s29.copyValueOf(c2, 3, 2)); //de

int a=35;
System.out.println(a); //35
System.out.println(String.format("%d", a)); //35
System.out.println(String.format("|%d|", a)); //|35|
System.out.println(String.format("|%5d|", a)); //5- total width | 35|
System.out.println(String.format("|%-5d|", a)); //|35 |
System.out.println(String.format("|%06d|", a)); //|000035|
//System.out.println(String.format("|%-06d|", a)); //exception
System.out.println(String.format("|%f|", 123.456)); //|123.456000|
System.out.println(String.format("|%.2f|", 123.456)); //|123.46|
System.out.println(String.format("|%8.2f|", 123.456)); //| 123.46|
}
}

```

4. StringBuffer class

- present in java.lang.*
- It is a mutable class, variable length of char (ie) we can increase or decrease its size at runtime
- By default StringBuffer is synchronized or thread safe, it always gives lower performance
- Default capacity of StringBuffer is 16
- We can't override equals() in StringBuffer, using toString() we convert StringBuffer to String class and then we apply equals() - it returns false

Constructors

1. StringBuffer()
2. StringBuffer(String s)
3. StringBuffer(int capacity)

Methods

1. int length()
2. char charAt(int pos)
3. void setCharAt(int index,char c)
4. void setLength(int length)
5. int capacity()
6. StringBuffer append(int i)
 - StringBuffer append(char c)
 - StringBuffer append(String s)
 - add at the end of StringBuffer
7. StringBuffer insert(int index,int i)
 - StringBuffer insert(int index,char i)
 - StringBuffer insert(int index,String i)
 - insert anywhere in StringBuffer
8. StringBuffer reverse()
9. StringBuffer replace(int start,int end, String s) //start to end-1
10. StringBuffer delete(int start,int end) - delete group of char
11. StringBuffer deleteCharAt(int index) - delete single char

```
public class Main {
    public static void main(String[] args) {
        StringBuffer sb1=new StringBuffer("Hello");
        System.out.println(sb1); //Hello
        System.out.println(sb1.length()); //5
        System.out.println(sb1.capacity()); //21 = 16+5
        System.out.println(sb1.charAt(1)); //e
        sb1.setCharAt(1, 'i');
        System.out.println(sb1); //Hillo
        sb1.setLength(2);
        System.out.println(sb1); //Hi
        System.out.println(sb1.length()); //2

        int b=10;
        StringBuffer sb2=new StringBuffer();
        String s=sb2.append("a=").append(b).append("!").toString();
        System.out.println(s); //a=10!

        StringBuffer sb3=new StringBuffer("I Java");
        System.out.println(sb3.insert(2, "like ")); //I like Java

        StringBuffer sb4=new StringBuffer("Hello");
        System.out.println(sb4.reverse()); //olleH

        StringBuffer sb5=new StringBuffer("This is a test");
        System.out.println(sb5.replace(5, 7, "was")); //This was a test

        StringBuffer sb6=new StringBuffer("This is a test");
        System.out.println(sb6.delete(5, 7)); //This a test
        System.out.println(sb6.deleteCharAt(0)); //his a test
    }
}
```

5. StringBuilder class

- Available from JDK1.5 onwards
- It is similar to StringBuffer but it is not synchronized or threadsafe so it gives better performance

From: Senthil Kumar T
Sent: Monday, September 4, 2023 3:30 PM
To: Yerramsetti V Subrahmanyamaisatyanarayana <yerramsetti.v.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalaran.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tenguna Vishnu Vardhan Rao <tenguna.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Darsh Srivastava <adarsh-srivastava@hcl.com>; Tava Durga <tava.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajednra <aherrutuja.rajednra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at <hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; Kohale Siddhesh Prashant <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant.shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashank Sanjay Deshmene <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afried <shaik.afried@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settippalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagy Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakuja <sanidhya.sakuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shaminis <shaminis@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddii <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahul.venkatan.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavarapu <priyanka.juijavarapu@hcl.com>; Priyadarshini M <m.priyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r_priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya

Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur.gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarthi.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanmant@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin_k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoormv@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <pandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevanya.mondh@hcl.com>; Aravind V <aravind_v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth_l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <MonishaI@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhya <arolla.anuhya@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonza@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.a@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапala <[ashini.rathnапала@hcl.com](mailto:ashini.rathnапala@hcl.com)>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha_sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kolivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore_p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kavya J <kavya.j@hcl.com>; Kavitha S <kavitha_s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakam.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchen Gurunath <kanchen.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil_kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.maland@hcl.com>; Kalpana Golla <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kadke Amrita Bhujangrao <kadke.amrita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K P Argunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judson.sam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijaumallikar.lohar@hcl.com>; Jelio Hallies Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitteja@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda

Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendaran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Dontu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashs@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <daluvai.hemalatha@hcl.com>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedula Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 5 Notes

Classification: **Confidential**

Handson

1. Team Details

Create a class named Team with the following private member variables / attributes

```
String name;
String coach;
String location;
String players
String captain
```

Include appropriate getters and setters.

[Naming Convention:
getters : getName getCoach ..
setters : setName, setCoach...]

Include a 5-argument argument constructor in this class. The arguments passed to the constructor are in this order --- name, coach, location,players,captain.

Include a default empty constructor.

Create another class named Main and include a main method to test the above class(Print the output in Main Class).

Input and Output Format:

Refer sample input and output for formatting specifications.

Read the team details as a string value separate by '#'. Use string.split() function to display the team details

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the team details

CSK#Stephen Fleming#Chennai#MS Dhoni,Aswin,Raina,Hussey,Maxwel,Bravo,Morkel,Jadeja,Mohit Sharma,Hayden,du plessis,Abhinav Mukund#MS Dhoni

Team : CSK

Coach : Stephen Fleming

Location : Chennai

Players : MS Dhoni,Aswin,Raina,Hussey,Maxwel,Bravo,Morkel,Jadeja,Mohit Sharma,Hayden,du plessis,Abhinav Mukund

Captain : MS Dhoni

2. Wicket details

Create a class named Wicket with the following private member variables / attributes

Long over
Long ball

```
String wicketType
String playerName
String bowlerName
```

Include a 5-argument argument constructor in this class. The arguments passed to the constructor are in this order --- over, ball,wicketType,playerName,bowlerName.

Include a default empty constructor.

Include appropriate getters and setters.

[Naming Convention:

```
getters : getOver getBall ...
setters : setOver, setBall...]
```

Create another class named Main and include a main method to test the above class(Print the output in Main Class).

Input and Output Format:

Refer sample input and output for formatting specifications.

Use array of objects to read wicket details and use String.split() function to display the wicket details.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the number of wickets

3

Enter the details of wicket 1

5,2,LBW,Gautam Gambir,Aswin

Enter the details of wicket 2

6,6,Bowled,Brad Hogg,Dwayne Bravo

Enter the details of wicket 3

7,3,Stumped,Robin Uthappa,Ravindra Jadeja

Wicket Details

Over : 5

Ball : 2

Wicket Type : LBW

Player Name : Gautam Gambir

Bowler Name : Aswin

Over : 6

Ball : 6

Wicket Type : Bowled

Player Name : Brad Hogg

Bowler Name : Dwayne Bravo

Over : 7

Ball : 3

Wicket Type : Stumped

Player Name : Robin Uthappa

Bowler Name : Ravindra Jadeja

Predefined Packages

1. java.lang.* - Language Package - only one optional pkg

1. Object class

- super class of all the classes (ie) any predefined or userdefined class by default inherit Object class

void add(int a) - int as arg

void add(String s)- String as arg

void add(Test t) - object of only Test class as arg

void add(Employee e) - object of only Employee class as arg

void add(Object o) - object of any class as arg

Constructor

- Object()

Methods

1. String **toString()** - String representation of an object

```
String s1=new String("Hello"); //all predefined class by default contains toString(), so if we print object of any predefined class it will print the content instead of object reference
sop(s1); //Hello
```

Sample s2=new Sample("Hello"); //Userdefined

```
sop(s2); //object reference
```

Whenever we print object of any class it will print only memory reference. But we don't print memory reference instead we want to print some content then in that case we have to override `toString()`

```
class A {
    String s1;
    A(String s){
        s1=s;
    }
    @Override
    public String toString() {
        return s1;
    }
}

public class Main {
    public static void main(String[] args) {
        String a1=new String("Hello");
        System.out.println(a1); //Hello
        A a2=new A("Hello");
        //System.out.println(a2); //object reference
        System.out.println(a2); //automatically invokes toString() - Hello
        System.out.println(a2.toString()); //Hello
    }
}
```

2. `boolean equals(Object o)` - used to check equality of the content
3. `==` (equals versus) - used to check equality of object reference

If we are not using `new` operator, then `==` also works like `equals()`

```
public class Main {
    public static void main(String[] args) {
        Integer a1=new Integer(42);
        Integer a2=new Integer(42);
        System.out.println(a1.equals(a2)); //true
        System.out.println(a1==a2); //false
        String s1=new String("Hello");
        String s2=new String("Hello");

        System.out.println(s1.equals(s2)); //true
        System.out.println(s1==s2); //false
        String s3=s2;
        System.out.println(s2.equals(s3)); //true
        System.out.println(s2==s3); //true
        String s4="Hello"; //literal
        String s5="Hello"; //literal
        System.out.println(s4.equals(s5)); //true
        System.out.println(s4==s5); //true
    }
}
```

4. `int hashCode()`

- When we store some value into object, internally an address will be generated for that value, to return that address we use `hashCode()`
- If two objects are equal according to `equals()`, then `hashCode` will return same

```
public class Main {
    public static void main(String[] args) {
        String s1=new String("Hello");
        String s2=new String("Hello");
        System.out.println(s1.hashCode());
        System.out.println(s2.hashCode());
    }
}
```

5. `protected void finalize()`

`Box b=new Box();` real object stored in heap
reference(stack)

```
b=null;
```

Explicitly Garbage Collection

1. System.gc()
2. Runtime.gc()

finalize() used to perform cleanup activity before destroying the object. It is called by garbage collector before destroying the object from memory and invoked only once in the lifecycle of program

Unreferencing the object

1. By anonymous object

Anonymous object are those object which are created without reference variable

```
new Box().add();
```

2. By nulling reference

```
Box b=new Box();
b=null;
```

3. By assigning the reference to another variable

```
Box b1=new Box(); //1000
Box b2=new Box(); //2000
b1=b2;
sop(b1); //2000
```

1. finalize() of which class is going to be called

```
public class Main {
    public static void main(String[] args) {
        String s1=new String("Hello");
        s1=null;
        System.gc();
        System.out.println("Garbage Collection is done");
    }
    protected void finalize() {
        System.out.println("Cleanup is done");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Main s1=new Main();
        s1=null;
        System.gc();
        System.out.println("Garbage Collection is done");
    }
    protected void finalize() {
        System.out.println("Cleanup is done");
    }
}
```

2. Explicit call to finalize

```
public class Main {
    public static void main(String[] args) {
        Main s1=new Main();
        Main s2=new Main(); //2000
        s1=s2;
        s1.finalize(); //not destroy the object
        System.out.println("Garbage Collection is done");
        System.gc();
    }
    protected void finalize() {

        System.out.println("Cleanup is done");
    }
}
```

6. final void wait()
7. final void wait(long msec)
8. final void notify()
9. final void notifyAll()
10. Object clone()
 - used to create exact copy of an object
 - clone() always follow shallow copy
 - If we want to take copy of an object using clone(), then that class should implements Cloneable interface which is a marker interface, in case if it is not implemented Cloneable interface then it will raise CloneNotSupportedException

Shallow copy

- will have exact copy of all fields of original object (ie) any changes made to those object through clone object will be reflected in original object and vice versa

```
class Address {
    String city;
    String state;
    String country;
    public Address(String city, String state, String country) {
        super();
        this.city = city;
        this.state = state;
        this.country = country;
    }
}
class Employee implements Cloneable {
    int empid;
    String name;
    Address address;
    public Employee(int empid, String name, Address address) {
        super();
        this.empid = empid;
        this.name = name;
        this.address = address;
    }
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }
}
public class Main {
    public static void main(String[] args) {
        Address addr=new Address("Chennai","Tamilnadu","India");
        Employee e1=new Employee(10,"Ram",addr);
        Employee e2=null;
        try {
            e2=(Employee)e1.clone();
        } catch (CloneNotSupportedException e) {
            e.printStackTrace();
        }
        System.out.println(e1.address.state); //Tamilnadu
        e2.address.state="Kerala";
        System.out.println(e1.address.state); //Kerala
    }
}
```

2. Wrapper class

- classes that supports the primitive datatype, to perform any operation on datatype
- All wrapper classes are immutable class(u cant change)

```
int a=10;
```

datatype wrapper class
 int - Integer
 byte - Byte
 short - Short
 long - Long
 boolean - Boolean
 char - Character
 float - Float
 double - Double

1. Float class
 - used to perform operation on float datatype

Constructor

1. `Float(float f)`
`Float f1=new Float(3.14f);`
2. `Float(double d)`
`Float f2=new Float(3.14);`
3. `Float(String s) throws NumberFormatException`
`Float f3=new Float("3.14");`
`Float f4=new Float("1bc"); //NumberFormatException`

```
int a=10, b=10;
if(a==b)
```

```
float f1=3.14f,f2=3.14f;
if(f1==f2) //error
```

Methods

1. `static int compare(float f1,float f2)` - compare 2 float datatype, if equal it return 0, if greater 1, if lesser -1
2. `int compareTo(Float f)` - compare 2 Float object , if equal it return 0, if greater 1, if lesser -1
3. `static boolean isInfinite()` - check float datatype is infinite or not
4. `boolean isInfinite()` - check Float object is infinite or not
5. `static boolean isFinite()` - check float datatype is finite or not
6. `boolean isFinite()` - check Float object is finite or not
7. `static boolean isNaN()` - check float datatype is nan or not
8. `boolean isNaN()` - check Float object is nan or not
9. `static float parseFloat(String s) throws NFE` - convert String to related datatype
10. `static Float valueOf(String s) throws NFE` - convert String to Float object
11. `float floatValue()` - convert Float object to float datatype
12. `int intValue()`
13. `double doubleValue()`
14. `short shortValue()`
15. `long longValue()`

Constants

1. `Float.MAX_VALUE`
2. `Float.MIN_VALUE`
3. `Float.POSITIVE_INFINITY`
4. `Float.NEGATIVE_INFINITY`
5. `Float.TYPE`

```
public class Main {
    public static void main(String[] args) {
        float f1=3.14f;
        float f2=3.14f;
        System.out.println(Float.compare(f1,f2)); //0

        Float f3=new Float(3.14f);
        Float f4=new Float(3.14f);
        System.out.println(f3.compareTo(f4)); //0

        float f5=(float)(1/0.);
        System.out.println(Float.isInfinite(f5)); //true
        Float f6=new Float(1/0.);
        System.out.println(f6.isInfinite()); //true

        float f7=(float)Math.sqrt(-4);
        System.out.println(FloatisNaN(f7)); //true
        Float f8=new Float(Math.sqrt(-4));
        System.out.println(f8.isNaN()); //true

        String s="3.14";
        float f9=Float.parseFloat(s);
        System.out.println(f9); //3.14
        Float f10=Float.valueOf(s);
        System.out.println(f10); //3.14
        float f11=f10.floatValue();
        System.out.println(f11); //3.14
        int i=f10.intValue();
```

```

        System.out.println(i); //3

        System.out.println(Float.MAX_VALUE);
        System.out.println(Float.MIN_VALUE);
        System.out.println(Float.POSITIVE_INFINITY);
        System.out.println(Float.NEGATIVE_INFINITY);
        System.out.println(Float.TYPE);
    }
}

```

2. Double class

- used to perform operation on double datatype

Constructor

- Double(double d)
- Double(String s) throws NFE

From: Senthil Kumar T

Sent: Friday, September 1, 2023 3:25 PM

To: Yerramsetti V Subrahmanyam Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Amer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; 'Aastha .' <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalakshmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek-ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarakumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajendran <aherrutuja.rajendra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhashish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; sridhar.at@hcl.com; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteek <akasam.karteek@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antom@hcl.com>; Kohale Siddhesh Prashant <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth_kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant_shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashank Sanjay Deshmane <shashanksan.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <ShanmugaPriyaK@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellula <shaikibrahim.khale@hcl.com>; Shaik Afrid <shaik.afrid@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagyachan Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <saraswathi.renga@hcl.com>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandu <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sakuja <sanidhya.sakuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila

Rangana <samila.rangana@hcl.com>; Sameer . <sameer_s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit_kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamini <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepelli <kallepelli.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan_pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; ricardo.lopezporras@hcl.com; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddri <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andhareartibaban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; rahulvenkat.gone@hcl.com; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar-rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja_singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Jujiavarapu <priyanka.jujiavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r_priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratikthakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasah.deshm@hcl.com>; Prathiyunnan R <prathiyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant_gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagaral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khatti <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur.gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanma@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padva Premsingh <padva.premsingh@hcl.com>; Pachimatl Laya <pachimatl.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin_sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh_anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita_n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyashree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevana.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; 'Archana Thineta' <archana.thineta@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monishal@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyaa <arolla.anuhyaa@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <mamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumar.aggarwal@hcl.com>; M Hemalatha <hemalatha@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnapala <ashini.rathnapala@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhuri Kunal Prabhakar <chaudhuri.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwary@hcl.com>; Bhosalekshiti Somes <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritikas@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondouji Srinidhi <kondouji.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata_satya@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore.p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha.s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakan.abhijit@hcl.com>; Kasipata Samarpiana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>;

Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumartr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil.kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; kalpana.golla@hcl.com; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunka@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judson.sam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jijau.mallikar.lohar@hcl.com>; Jelio Halleys Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <p.janani@hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadurai S <jabadurai.s@hcl.com>; Ishita Raj <ishitaraj.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit_mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saituja@hcl.com>; Gottipati Nikhitha <gottipati_nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Varas Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendar <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja_b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Dontu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati_bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandepalli Naveen <gandepalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuva@hcl.com>; Eliezer David Gutierrez Hinjosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>; Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashs@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david-lynch@hcl.com>; Dasthuri Ramya <dasthuri.ramya@hcl.com>; Daluvai Hemalatha <[daluvai.hemalatha@hcl.com](mailto:daluvi.hemalatha@hcl.com)>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimireddy Praveen Kumar <chitimireddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedula Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 4 Notes

Classification: **Confidential**

Handson

1. Abstract Class I – Shape

[Adhere to the OOPs specifications specified here. Follow the naming conventions for getters and setters. Download the template code provided and fill in the missing code]

Create an abstract class named Shape with the following protected attributes / member variables.

String name

Include a 1-argument constructor.

Include getters and setters.

Include an abstract method named calculateArea() . This method returns a Float value.

Create a class named Circle . The class Circle is a derived class of Shape. Include the following private attributes / member variables.

Integer radius

Include a 2-argument constructor. The order of the arguments is name, radius.

Include getters and setters.

Override the abstract method calculateArea() defined in the Shape class. This method returns the area of the circle. [Take the value of pi as 3.14]

Create a class named Square . The class Square is a derived class of Shape. Include the following private attributes / member variables.

Integer side

Include a 2-argument constructor. The order of the arguments is name, side.

Include getters and setters.

Override the abstract method calculateArea() defined in the Shape class. This method returns the area of the square.

Create a class named Rectangle . The class Rectangle is a derived class of Shape. Include the following private attributes / member variables.

Integer length

Integer breadth

Include a 3-argument constructor. The order of the arguments is name, length, breadth

Include getters and setters.

Override the abstract method calculateArea() defined in the Shape class. This method returns the area of the rectangle.

Create another class called Main. In the method, create instances of the above classes and test the above classes.

Input and Output Format:

Refer sample input and output for formatting specifications.

All Float values are displayed correct to 2 decimal places.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output 1:

Circle

Square

Rectangle

Enter the shape name

Circle

Enter the radius

25

Area of Circle is 1962.50

Sample Input and Output 2:

Circle

Square

Rectangle

Enter the shape name

Square

Enter the side

23

Area of Square is 529.0

Sample Input and Output 3:

Circle

Square

Rectangle

Enter the shape name

Rectangle

Enter the length

45

Enter the breadth

60

Area of Rectangle is 2700.00

2. Abstract Class II – Card

[Adhere to the OOPs specifications specified here. Follow the naming conventions for getters and setters. Download the template code provided and fill in the missing code]

Create an abstract class named Card with the following protected attributes / member variables.

String holderName;

String cardNumber;

String expiryDate;

Include appropriate getters and setters.

Include appropriate constructors. In the 3-argument constructor, the order of the arguments is holderName, cardNumber, expiryDate.

Create a class named MembershipCard. The class MembershipCard is a derived class of Card. Include the following private attributes / member variables.

Integer rating

Include appropriate getters and setters.

Include appropriate constructors. In the 4-argument constructor, the order of the arguments is holderName, cardNumber, expiryDate, rating.

Create a class named PaybackCard. The class PaybackCard is a derived class of Card. Include the following private attributes / member variables.

Integer pointsEarned;

Double totalAmount;

Include appropriate getters and setters.

Include appropriate constructors. In the 5-argument constructor, the order of the arguments is holderName, cardNumber, expiryDate, pointsEarned, totalAmount.

Create another class called Main. In the method, create instances of the above classes and test the above classes. The card details are entered separated by a '|'.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output 1:

Select the Card

1.Payback Card

2.Membership Card

1

Enter the Card Details:

Anandhi|12345|14/01/2020

Enter points in card

1000

Enter Amount

50000

Anandhi's Payback Card Details:

Card Number 12345

Points Earned 1000

Total Amount 50000.0

Sample Input and Output 2:

Select the Card

1.Payback Card

2.Membership Card

2

Enter the Card Details:

Collin|45678|20/11/2021

Enter rating in card

10

Collin's Membership Card Details:

Card Number 45678

Rating 10

3. Simple Interface

An Interface consists of a contract of set of methods with only declaration and no implementation. Any class which implements the interface commits that all methods would be implemented.

Here is a program to illustrate a simple interface.

1. Create an interface IPlayerStatistics

Add a method with the following prototype

--- public void displayPlayerStatistics

2. Create class Player which implements the IPlayerStatistics Interface

Include private data members :

String name

String teamName

Integer noOfMatches

Long totalRunsScored

Integer noOfWicketsTaken

Include an 5 argument constructor with following arguments: name, teamName, noOfMatches, totalRunsScored, noOfWicketsTaken and implement the interface method public void displayPlayerStatistics to display the player details.

Create a Main class with main method to test test above classes.

Sample Input and Output 1:

[All text in bold corresponds to input and the rest corresponds to output.]

Enter player name

Ravichandran Ashwin

Enter team name

Chennai Super Kings

Enter number of matches played

86

Enter total runs scored

185

Enter number of wickets taken

89

Player Details

Player name : Ravichandran Ashwin

Team name : Chennai Super Kings

No of matches : 86

Total runsscored : 185

No of wickets taken : 89

4.Interface II

1. Create an interface IPlayerStatistics

Add a method with the following prototype

public void displayPlayerStatistics

2. Create base class CricketPlayer

Include private data members : name, teamName, noOfMatches.

Include an 3 argument constructor with following arguments: name, teamName, noOfMatches.

3. Create derived class Bowler which extends CricketPlayer and implements the interface IplayerStatistics

Include private data members : noOfWickets.

Include an 1 argument constructor with the argument: noOfWickets and implement the interface method public void displayPlayerStatistics to display the player details.

4. Create derived class Batsman which extends CricketPlayer and implements the interface IPlayerStatistics

Include private data members : runs.

Include an 1 argument constructor with the argument: runs and implement the interface method public void displayPlayerStatistics to display the player details.

5. Create derived class WicketKeeper which extends CricketPlayer and implements the interface IPlayerStatistics

Include private data members : noOfCatches, noOfStumpings ,runs, noOfDismissals .

Include an 4 argument constructor with following arguments: noOfCatches, noOfStumpings,runs, noOfDismissals and implement the interface method public void displayPlayerStatistics to display the player details.

6. Create derived class AllRounder which extends CricketPlayer and implements the interface IPlayerStatistics

Include private data members : runs,noOfWickets.

Include an 2 argument constructor with following arguments: runs, noOfWickets and implement the interface method public void displayPlayerStatistics to display the player details.

Create a Main class with main method to test test above classes.

Input and Output Format:

Refer sample input and output for format specifications.
[All text in bold corresponds to input and the rest corresponds to output.]

Sample Input and Output 1:

Menu

- 1.Bowler
- 2.Batsman
- 3.WicketKeeper
- 4.AllRounder

Enter your choice

1

Enter the Bowler details

Enter player name

Ravichandran Ashwin

Enter team name

Chennai Super Kings

Enter number of matches played

111

Enter number of wickets taken

100

Player name : Ravichandran Ashwin

Team name : Chennai Super Kings

No of matches : 111

No of wickets taken : 100

Do you want to continue?

YES

Menu

- 1.Bowler
- 2.Batsman
- 3.WicketKeeper
- 4.AllRounder

Enter your choice

4

Enter the AllRounder details

Enter player name

Shane Watson

Enter team name

Royal Challengers Bangalore

Enter number of matches played

94

Enter the runs scored

2551

Enter number of wickets taken

81

Player name : Shane Watson

Team name : Royal Challengers Bangalore

No of matches : 94

Runs scored : 2551

No of wickets taken : 81

Do you want to continue?

NO

Sample Input and output 2:

Menu

- 1.Bowler
- 2.Batsman
- 3.WicketKeeper
- 4.AllRounder

Enter your choice

2

Enter the Batsman details

Enter player name

Virat Kohli

Enter team name

Royal Challengers Bangalore

Enter number of matches played

139

Enter the runs scored

4110

Player name : Virat Kohli

Team name : Royal Challengers Bangalore

No of matches : 139
 Runs scored : 4110
 Do you want to continue?
 YES
 Menu
 1.Bowler
 2.Batsman
 3.WicketKeeper
 4.AllRounder
 Enter your choice
 3
 Enter the WicketKeeper details
 Enter player name
 Mahendra Singh Dhoni
 Enter team name
 Chennai Super Kings
 Enter number of matches played
 143
 Enter number of catches taken
 52
 Enter number of stumpings
 22
 Enter number of dismissals
 74
 Enter the runs scored
 3271
 Player name : Mahendra Singh Dhoni
 Team name : Chennai Super Kings
 No of matches : 143
 No of catches taken : 52
 No of stumpings : 22
 No of dismissals : 74
 Runs scored : 3271
 Do you want to continue?
 NO

Problem Statement:

What is the output of the following program?

```
class Eggs {
    int doX(Long x, Long y) { return 1;}
    int doX(long... x) { return 2;}
    int doX(Integer x, Integer y) { return 3; }
    int doX(Number n, Number m) { return 4; }

    public static void main(String[ ] args) {
        new Eggs().go();
    }

    void go() {
        short s = 7;
        System.out.println(doX(s,s) + " ");
        System.out.println(doX(7,7));
    }
}
```

- A. 1 1
- B. 2 1
- C. 3 1
- D. 4 1
- E. 2 3
- F. 3 3
- G. 4 3

Problem Statement:

What is the output of the following program?

```
class Mixer {
    Mixer() {}
    Mixer(Mixer m) { m1 = m; }
    Mixer m1;
```

```

public static void main(String[ ] args) {
    Mixer m2 = new Mixer();
    Mixer m3 = new Mixer(m2);
    m3.go();
    Mixer m4 = m3.m1;
    m4.go();
    Mixer m5 = m2.m1;
    m5.go();
}
void go() {
    System.out.println("hi ");
}
}

```

Problem Statement:

What is the output of the following program?

```

class Bird {
    {System.out.print("b1 ");}
}

public Bird() {
    System.out.print("b2 ");
}
}

```

```

class Raptor extends Bird {
    static { System.out.print("r1 "); }

    public Raptor() {
        System.out.print("r2 ");
    }

    { System.out.print("r3 "); }

    static { System.out.print("r4 "); }
}

```

```

class Hawk extends Raptor {
    public static void main(String[] args) {
        System.out.print("pre ");
        new Hawk();
        System.out.println("hawk ");
    }
}

```

- A. pre b1 b2 r3 r2 hawk
- B. pre b2 b1 r2 r3 hawk
- C. pre b2 b1 r2 r3 hawk r1 r4
- D. r1 r4 pre b1 b2 r3 r2 hawk
- E. r1 r4 pre b2 b1 r2 r3 hawk
- F. pre r1 r4 b1 b2 r3 r2 hawk
- G. pre r1 r4 b2 b1 r2 r3 hawk
- H. The order of output cannot be predicted.
- I. Compilation fails.

Problem Statement:

What is the output of the following program?

```

class Clidders {
    public final void flipper() {
        System.out.println("Clidder");
    }
}

```

```

public class Clidlets extends Clidders {
    public void flipper() {
        System.out.println("Flip a Clidlet");
    }
}

```

```

        super.flipper();
    }
    public static void main(String[] args) {
        new Clidlets().flipper();
    }
}

```

- A. Flip a Clidlet
 B. Flip a Clidder
 C. Flip a Clidder
 Flip a Clidlet
 D. Flip a Clidlet
 Flip a Clidder
 E. Compilation fails.

Problem Statement:

Given the following:

- i. interface Base {
- ii. boolean m1();
- iii. byte m2(short s);
- iv. }

Which code fragments will compile? (Choose all that apply.)

- a) interface Base2 implements Base {}
- b) abstract class Class2 extends Base {
 public boolean m1() { return true; } }
- c) abstract class Class2 implements Base {}
- d) abstract class Class2 implements Base {
 public boolean m1() { return (true); } }
- e) class Class2 implements Base {
 boolean m1() { return false; }
 byte m2 (short s) { return 42; } }

Problem Statement:

Which of the following declare a compatible abstract class? (Choose all that apply.)

- a) public abstract class Canine { public Bark speak(); }
- b) public abstract class Canine { public Bark speak() {} }
- c) public class Canine { public abstract Bark speak(); }
- d) public class Canine abstract { public abstract Bark speak(); }

Problem Statement:

Given:

```
public abstract interface Froblicate { public void twiddle(String s); }
```

Which is a correct class?

- a) public abstract void twiddle(String s) {}
- b) public abstract class Frob implements Froblicate {}
- c) public class Frob extends Froblicate {
 public void twiddle(Integer i) {} }
- d) public class Frob implements Froblicate {
 public void twiddle(Integer i) {} }
- e) public class Frob implements Froblicate {
 public void twiddle(String i) {}
 public void twiddle(Integer s) {} }

Problem Statement:

Given:

- 1. class Zing {
- 2. protected Hmpf h;

3. }
4. class Woop extends Zing { }
5. class Hmpf { }

Which is true?

- A. Woop IS-A Hmpf and HAS-A zing.
- B. Zing IS-A Woop and HAS-A Hmpf.
- C. Hmpf HAS-A Woop and Woop IS-A Zing.
- D. Woop HAS-A Hmpf and Woop IS-A Zing.
- E. Zing HAS-A Hmpf and Zing IS-A Woop.

Problem Statement:

Given:

```
public class MyOuter {
    public static class MyInner {
        public static void foo() {} }
    }
}
```

Which, if placed in a class other than MyOuter or MyInner, instantiates an instance of the nested class?

- A. MyOuter.MyInner m = new MyOuter.MyInner();
- B. MyOuter.MyInner m2 = new MyInner();
- C. MyOuter m = new MyOuter();
- MyOuter.MyInner mi = m.new MyOuter.MyInner();
- D. MyInner mi = new MyOuter.MyInner();

Problem Statement:

What is the output of the following program?

```
public abstract class AbstractTest {

    public int getNum() {
        return 45;
    }

    public abstract class Bar {
        public int getNum() {
            return 38;
        }
    }

    public static void main(String[] args) {
        AbstractTest t = new AbstractTest() {
            public int getNum() {
                return 22;
            }
        };
        AbstractTest.Bar f = t.new Bar() {
            public int getNum() {
                return 57;
            }
        };
        System.out.println(f.getNum() + " " + t.getNum());
    }
}
```

- A. 57 22
- B. 45 38
- C. 45 57
- D. An exception occurs at runtime.
- E. Compilation fails.

super keyword

- It is used to access ur base class constructor, base class method and base class variable

```
class A {
    int i;
    int j;

    A(int a,int b){
        i=a;
```

```

        j=b;
    }

    void show() {
        System.out.println("i= "+i+" j= "+j);
    }
}
class B extends A {
    int k;

    B(int a,int b,int c){
        super(a,b);
        k=c;
    }
    void show() {
        super.show();
        System.out.println("k= "+k);
    }
}
public class Main {
    public static void main(String[] args) {
        B b=new B(1,2,3);
        b.show();
    }
}

```

Method overriding

- same method name, same number, order and datatype of arg, same return type of method and present in different class and the class should be inherited

No Constructor overriding**Dynamic Method Dispatch**

- In order to have effective method overriding, we go for Dynamic Method dispatch
- We always create an object for base class but we store the reference of derived class but not vice versa, so the compiler thinks that object is created for base class but only at runtime it comes to know it contains the reference of derived class, in this way we achieve runtime polymorphism

```

class Sphere {
    void volume() {
        System.out.println("Sphere volume");
    }
}
class Hemisphere extends Sphere {
    void volume() {
        System.out.println("Hemisphere volume");
    }
}
public class Main {
    public static void main(String[] args) {
        /*Hemisphere h=new Hemisphere();
        h.volume(); //Hemisphere volume
        Sphere s=new Sphere();
        s.volume();*/ //Sphere volume

        Sphere s=new Hemisphere(); //DMD
        s.volume(); //Hemisphere volume
        s=new Sphere(); //Hemisphere reference garbage collected and s contain sphere reference
        s.volume(); //Sphere volume
    }
}

//Hierachial inheritance
class A {
    void callback() {
        System.out.println("A's callback implementation");
    }
}
class B extends A {
    void callback() {
        System.out.println("B's callback implementation");
    }
}

```

```

    }
    class C extends A {
        void callback() {
            System.out.println("C's callback implementation");
        }
    }
    public class Main {
        public static void main(String[] args) {
            A a=new A();
            a.callback();
            a=new B(); //DMD
            a.callback();
            a=new C();
            a.callback();
        }
    }
}

```

How constructors are invoked in inheritance?

- Invoked as top down approach

```

//Multilevel inheritance
class A {
    A() {
        System.out.println("1");
    }
}
class B extends A {
    B() {
        System.out.println("2");
    }
}
class C extends B{
    C(){
        System.out.println("3");
    }
}
public class Main {
    public static void main(String[] args) {
        C c=new C(); //123
    }
}

```

```

//Multilevel inheritance
class A {
    A(int a) {
        System.out.println("1");
    }
}
class B extends A {
    B() {
        System.out.println("2");
    }
}
class C extends B{
    C(){
        System.out.println("3");
    }
}
public class Main {
    public static void main(String[] args) {
        C c=new C(); //compilation error
    }
}

```

First it will go to the related constructor (ie) into C class constructor and check whether we have used super() or this(), then it will work according to that, if it is not then by default it will invoke only default constructor of base class

```

//Multilevel inheritance
class A {
    A(int a) {

```

```

        System.out.println("1");
    }
    /*A() {
        System.out.println("4");
    }*/
}
class B extends A {
    B() {
        super(10);
        System.out.println("2");
    }
}
class C extends B{
    C(){
        System.out.println("3");
    }
}
public class Main {
    public static void main(String[] args) {
        C c=new C(); //123
    }
}

```

final keyword

- It is a non access specifier
- When a class is declared as final, inheritance is not possible

```

final class A {
    A() {
        System.out.println("1");
    }
}
class B extends A { //error
    B() {
        System.out.println("2");
    }
}
- When a method is declared as final, method overriding is not possible
class A {
    final void add() {
        System.out.println("1");
    }
}
class B extends A {
    void add() { //error
        System.out.println("2");
    }
}
- When a variable is declared as final, it will acts like constant

```

Relationship - 2 types

1. is a relationship - when we perform inheritance

```

class Animal{
}
class Cat extends Animal{
}

```

Cat is a Animal

2. has-a relationship - Composition - one class contains another class

```

class Wheel{
}
class Car{
    Wheel w=new Wheel(); //Car has a Wheel
}

```

Covariant return type

- Before JDK1.5, it is not possible to override any method by changing its return type
- From JDK1.5, it is possible to override method by changing its return type (ie) only non primitive (ie) object
- Avoid type casting, wrong typecasting which leads to ClassCastException at runtime

```

class A {
    A show() {
        return this;
    }
    void print() {
        System.out.println("Inside A's method");
    }
}
class B extends A {
    B show(){
        return this;
    }
    void print() {
        System.out.println("Inside's B method");
    }
}
class C extends B {
    C show(){
        return this;
    }
    void print() {
        System.out.println("Inside's B method");
    }
}
public class Main {
    public static void main(String[] args) {
        A a1=new A();
        a1.show().print(); //Inside A's method
        B b1=new B();
        b1.show().print(); //Inside B's method
    }
}

```

Compound block/Instance block

- It will be invoked before the constructor invokes

Syntax: {
 //logic
}

only in case inheritance, first all static block will be executed from top to bottom

18 21 19 1 6 11 15 17 5 7 2 4 3 8 10 9 14 16 12 13 20

```

class A {
    static { //static block
        System.out.println("1");
    }
    A(){
        System.out.println("2");
    }
    A(String s){
        this(10);
        System.out.println("3");
    }
    A(int a){
        this();
        System.out.println("4");
    }
    { //Compound block
        System.out.println("5");
    }
    static {
        System.out.println("6");
    }
    {
        System.out.println("7");
    }
}
class B extends A {

```

```

    {
        System.out.println("8");
    }
B(){
    super("hello");
    System.out.println("9");
}
{
    System.out.println("10");
}
static {
    System.out.println("11");
}
}
class C extends B {
    C(){
        System.out.println("12");
    }
    C(int a){
        this();
        System.out.println("13");
    }
    {
        System.out.println("14");
    }
    static {
        System.out.println("15");
    }
    {
        System.out.println("16");
    }
    static {
        System.out.println("17");
    }
}
public class Main {
    static {
        System.out.println("18");
    }
    public static void main(String[] args) {
        System.out.println("19");
        C c=new C(10);
        System.out.println("20");
    }
    static {
        System.out.println("21");
    }
}

```

abstract keyword

- It is a non access specifier
- When a class is declared as abstract, u cannot create an object for that class
- When a method is declared as abstract, it does not contain any definition, it just ends with semicolon
- Variables cant be declared as abstract
- When a class contains abstract method then that class should be declared as abstract otherwise error, but not necessarily all abstract class should contain abstract methods
- Abstract class can also non abstract method (normal method)
- Abstract class can also be inherited, so when u inherit an abstract class then compulsorily we have to provide the definition of abstract method in the inherited class or if we not defining the definition then we have to make inherited class itself as abstract
- Abstract class contain constructor but we cant invoke it

```

abstract class A {
    abstract void show();
    void method() {
        System.out.println("Concrete implementation");
    }
}
/*abstract*/ class B extends A {
    void show() {
        System.out.println("Abstract method");
    }
}

```

```

    }
public class Main {
    public static void main(String[] args) {
        B b=new B();
        b.show();
        b.method();
    }
}

```

Anonymous Inner class

- used to access normal method of abstract class without inheritance

```

abstract class A {
    abstract void show();
    void add() {
        System.out.println("Concrete implementation");
    }
}

```

```

public class Main {
    public static void main(String[] args) {
        A a=new A() { //AIC
            public void show() {
                System.out.println("abstract method");
            }
            public void add() {
                System.out.println("hello");
            }
        };
        a.show();
        a.add();
    }
}

```

Interface

- No multiple inheritance instead we use interface

```

class A extends B,C,D{ //error
}

```

- Interface are syntactically similar to classes but contains only method declaration and variable declaration and initialization

Syntax:

```

<<accessspecifier>> interface interfacename {
    //method declaration
    //variable declaration and initialization
}

```

- By default all interfaces are abstract, so we can't create an object for interface
- By default all interface methods are public and abstract, so interface method doesn't contain any definition it just ends with semicolon
- By default all interface variables are public, final, static, so we can access interface variable using "interfacename.variablename"
- Using "implements" keyword we can use interface in class
- When a class implements an interface then compulsorily we have to provide the definition of interface method in the class with public access specifier or define the class itself to be abstract
- The implemented class can also contain non-interface methods
- Interface can also be inherited

1 class extends 1 class

1 class implement n interface

1 interface extends n interface

- Marker interface - interface which doesn't contain anything like Comparable, Serializable, Remote
- From JDK 1.8 onwards, interface can also contain default and static methods
- From JDK 1.9 onwards, interface can also contain private and private static methods

```

interface Maths {
    void arithmetic(int a,int b);
    int c=10;
}
/*abstract*/ class A implements Maths {
    @Override
    public void arithmetic(int a, int b) {

```

```

        System.out.println(a+b);
        System.out.println(Maths.c);
    }
}

class B implements Maths {
    @Override
    public void arthimetic(int a, int b) {
        System.out.println(a-b);
    }
}
public class Main {
    public static void main(String[] args) {
        /* A a=new A();
        a.arthimetic(5, 3); //8 10
        B b=new B();
        b.arthimetic(5, 1); */ //4

        Maths m=new A(); //DMD
        m.arthimetic(5, 3); //8 10
        m=new B();
        m.arthimetic(5, 1); //4
    }
}

interface A {
    void meth1();
}
interface B {
    void meth2();
}
interface C extends A,B{
    void meth3();
}
class Example implements C {

    @Override
    public void meth1() {
        System.out.println("Meth1");
    }

    @Override
    public void meth2() {
        System.out.println("Meth2");
    }

    @Override
    public void meth3() {
        System.out.println("Meth3");
    }
}

public class Main {
    public static void main(String[] args) {
        Example e=new Example();
        e.meth1();
        e.meth2();
        e.meth3();
    }
}

```

From: Senthil Kumar T**Sent:** Monday, August 28, 2023 3:30 PM

To: Yerramsetti V Subrahmanyam Saisatyanarayana <yerramsettiv.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; Aastha . <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh

Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalar.koppu@hcl.com>; Venkatesh S <svenkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Bala Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Yamshi Krishna Bitla <yamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ananda@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwal.arun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <twinkle_t@hcl.com>; Abhishek Ranjan <abhishek.ranjan@hcl.com>; Abhishek Saxena <AbhishekSax@hcl.com>; Tungena Vishnu Vardhan Rao <tungena.vishn@hcl.com>; Abhishek Singh <singh.abhishek@hcl.com>; Troshan Mihindu <troshan.mihindu@hcl.com>; Tripurari. Kumar Ray. <tripurarkumar.ray@hcl.com>; Tirumala Ketineni <tirumala.ketineni@hcl.com>; Thimisha Samarasinghe <thimisha.samar@hcl.com>; Tharun Birru <tharun.birru@hcl.com>; Thanneeru Syam Kumar <thanneerusyam.kumar@hcl.com>; Tejaswini Channe <tejaswini.channe@hcl.com>; Teena Chawla <teena.c@hcl.com>; Adarsh Srivastava <adarsh-srivastava@hcl.com>; Tavva Durga <tavva.durga@hcl.com>; Tanuj Agrawal <tanuj.agrawal@hcl.com>; Tamilvanan S <tamilvanan_s@hcl.com>; Swetha M <m.swetha@hcl.com>; Swathi U <swathi.u@hcl.com>; Swaraj Rawte <rawte.swarajmadhukar@hcl.com>; Duvva Swapna <duvva.swapna@hcl.com>; Suwetha G <suwetha.g@hcl.com>; Sushmitha Soundarajan <sushmitha.sound@hcl.com>; Aditya Kumar <adityakumar@hcl.com>; Surrya A P <surrya.ap@hcl.com>; Suresh Babu Seelam <sureshbabu.seelam@hcl.com>; Suraj Kumar Singh <surajkumar.singh@hcl.com>; Supriya Gavande <supriya.gavande@hcl.com>; Sunita Meti <sunita.meti@hcl.com>; Sunil Kumar Devarakonda <sunilkumar.devar@hcl.com>; Sumithra S <sumithra_s@hcl.com>; Suma M <suma.m@hcl.com>; Sudhanshu Raj <sudhanshu.raj@hcl.com>; Sudhakar Soragari <sudhakar.soragari@hcl.com>; Sudha Rani <sudha_rani@hcl.com>; Aher Rutuja Rajedhra <aherrutuja.rajedhra@hcl.com>; Sudarshani Perera <sudarshani.perera@hcl.com>; Aishani Basu <aishani.basu@hcl.com>; Subhasish Biswal <subhasish.b@hcl.com>; Srikanth Salumuri <srikanth.salumuri@hcl.com>; Ajay Atole <ajay.atole@hcl.com>; 'sridhar.at@hcl.com' <sridhar.at@hcl.com>; Sreenivasulu Avula <sreenivasulu.avula@hcl.com>; Ajay Prasad Verma <ajay.prasadverma@hcl.com>; Sreedevan Gopinath <sreedevan.gopinath@hcl.com>; Souvik Adhikary <souvik.adhikary@hcl.com>; Ajith E <ajith.e@hcl.com>; Soundarya Sriram <soundarya.sriram@hcl.com>; Soumyadeep Ghosh <soumyadeep.ghosh@hcl.com>; Sonal Pandey <sonal.pandey@hcl.com>; Somya Sinha <somya.sinha@hcl.com>; Somanath Pradhan <somanath.pradhan@hcl.com>; Akasam Karteeck <akasam.karteeck@hcl.com>; Akash Behune <akash.behune@hcl.com>; Sitharaj Seenivasan <sitharaj.seenivasan@hcl.com>; Siranjeevi Chandran <siranjeevi.chandran@hcl.com>; Sinthu P <sinthu.p@hcl.com>; Sindhya Sivasami <sindhya.sivasami@hcl.com>; Akash Prakashchandra Mujavadiya <akashprakashc.mujav@hcl.com>; Simhadri Vinod <simhadri.vinod@hcl.com>; Silvius Daniel Antohe Multescu <silviudaniel.antoh@hcl.com>; Kohale Siddhesh Prashant <kohalesiddhesh.prash@hcl.com>; Siddharth Kumar <siddharth.kumar@hcl.com>; Akhilesh . <akhilesh-a@hcl.com>; Shuhail Rahman <shuhail.rahaman@hcl.com>; Shubham Raj <shubham.raj@hcl.com>; Shruti Sinha <shruti.sinha@hcl.com>; Shruti Saxena <shruti.saxena@hcl.com>; Shrikant Shelar [External] <shrikant.shelar@hcl.com>; Shivani Nimmala <shivani.nimmala@hcl.com>; Shinde Nikita Vijaykumar <shindenikita.vijay@hcl.com>; Sheeba Sanopher Mohamed Mydeen <sheebasanopher.moham@hcl.com>; Akshita . <akshita.a@hcl.com>; Shashanki Sanjay Deshmame <shashankisanj.deshm@hcl.com>; Akul Singh <akul.singh@hcl.com>; Akula Chandini Neeraja <akula.chand@hcl.com>; Shanmugam Samandhavada Sundaram <shanmugam.saman@hcl.com>; Shanmuga Priya K <shanmugaPriyak@hcl.com>; Shailesh Kumar Yadav <shaileshkumar.yadav@hcl.com>; Shaik Usman Basha <shaik.usmanbasha@hcl.com>; Shaik Shafi <shaik.shafi@hcl.com>; Alberto Cruz Sierra <alberto.cruzsierra@hcl.com>; Shaik Ibrahim Khaleellu <shaikibrahim.khale@hcl.com>; Shaik Afrix <shaik.afrix@hcl.com>; Shabana Sheik <shabana.sheik@hcl.com>; Settipalli Shasikanth Reddy <settipalli.shasi@hcl.com>; Alok Kumar Srivastava <alokkumar.srivastava@hcl.com>; Selvamoorthy Selvam <selvamoorthy.selvam@hcl.com>; Seera Kusuma Kumari <seerakusuma.kumari@hcl.com>; Sebastian Nae <sebastian.nae@hcl.com>; Sayooj P <sayooj.p@hcl.com>; Savitha M <savitha-m@hcl.com>; Saubhagya Chandra Patri <saubhagyachan.patri@hcl.com>; Satish Yedida <satish.yedida@hcl.com>; Satheesh Kumar S <satheesh.kumars@hcl.com>; Sasikumar Balasubramani <sasikumar.balas@hcl.com>; Saravanan Sivakumar <saravanan.sivakumar@hcl.com>; Saraswathi Renganathan <[sarасwathi.renga@hcl.com](mailto:saraswathi.renga@hcl.com)>; Aman Kumar Raghav <amankumar.raghav@hcl.com>; Sapkal Rutuja Pandurang <sapkalrutuja.pandu@hcl.com>; Santosh Laxmanrao Katkar <santoshlaxman.katka@hcl.com>; Sanket Garg <sanket.garg@hcl.com>; Sanjay Kumar Prasad <sanjaykumar.prasad@hcl.com>; Sanidhya Sahuja <sanidhya.sahuja@hcl.com>; Sandeep Sirikonda <sandeep.sirikonda@hcl.com>; Samila Rangana <samila.rangana@hcl.com>; Sameer . <sameer.s@hcl.com>; Saktipada Upadhyaya <saktipada.u@hcl.com>; Sakshi Singh <sakshisi@hcl.com>; Sakshi Dhoke <sakshi.dhoke@hcl.com>; Amit Kashyap <amit.kashyap@hcl.com>; Sajja Chenchu Sai <sajja.chenchusai@hcl.com>; Sahan Dias <sahan.dias@hcl.com>; Sachin Rai <sachin.rai@hcl.com>; Sachin Marwha <sachin.marwha@hcl.com>; S Shamin <shamini.s@hcl.com>; S Megala <megalas@hcl.com>; Rutika Hemantrao Gaurkar <rutikahemantr.gaurk@hcl.com>; Rupesh Kallepellu <kallepellu.rupesh@hcl.com>; Rudra Halder <rudra.halder@hcl.com>; Rowbal A <rowbal.a@hcl.com>; Rohit Kumar <rohit.kum@hcl.com>; Rohan Pandey <rohan.pandey@hcl.com>; Rijwana Parveen Vakkapella <rijwanaparveen.vakka@hcl.com>; 'ricardo.lopezporras@hcl.com' <ricardo.lopezporras@hcl.com>; Raviteja Natta <raviteja.natta@hcl.com>; Amruta Thakre <amruta.thakre@hcl.com>; Ravi Kumar Vudayana <ravikumar.vudayana@hcl.com>; Anagandula Sai Vardhan <anagandula.saiva@hcl.com>; Rasika Dilip Kamble <rasikadilip.kamble@hcl.com>; Rangarajan M <rangarajan.m@hcl.com>; Anand D <anand.d@hcl.com>; Ramu Doddi <ramu.doddi@hcl.com>; Ramisetti Siva Krishna <ramisetti.sivak@hcl.com>; Rajneesh Dixit <rajneesh.dixit@hcl.com>; Rajeshwari Badaradinni <rajeshwari.badar@hcl.com>; Rajesh Kethepalle <rajesh.kethepalle@hcl.com>; Rajesh G S <rajesh.gs@hcl.com>; Rajendra Kumar Shihare <Rajendras@hcl.com>; Rajeev Mishra <rajeev.mishra@hcl.com>; Rajaram M <rajaram.m@hcl.com>; Andhare Arti Baban <andharearti.baban@hcl.com>; Andi Patchamalai <andi.patchamalai@hcl.com>; Raj Mishra <raj.mishra@hcl.com>; 'rahulvenkat.gone@hcl.com' <rahulvenkat.gone@hcl.com>; Rahul Sontakke <rahul.sontakke@hcl.com>; Rahul Kumar Mishra <rahulkumar.mishra@hcl.com>; Rahul Kumar <kumar.rahu@hcl.com>; Puneet Gupta <Puneet.G@hcl.com>; Puja Singh <puja.singh@hcl.com>; Priyanshu Nagar <priyanshu.nagar@hcl.com>; Priyanka Juijavarapu <priyanka.juijavarapu@hcl.com>; Priyadarshini M <mpriyadarshini@hcl.com>; Anisetti Anusha <anisetti.anusha@hcl.com>; Priya R <r.priy@hcl.com>; Pravin N <pravin.n@hcl.com>; Pratik Thakare . <pratik.thakare.p@hcl.com>; Pratik Babasaheb Deshmukh <pratikbabasaheb.deshm@hcl.com>; Prathyunnan R <prathyunnan.r@hcl.com>; Prateek Bansal <prateek.bansal@hcl.com>; Prashanthi Gunda <prashanthi.gunda@hcl.com>; Prashant Gupta <prashant.gup@hcl.com>; Prasad Bawankule <prasad.bawankule@hcl.com>; Pramod Nagral <pramod.nagaral@hcl.com>; Prakash Raj K <prakashraj.k@hcl.com>; Prafull Chandrakant Borchate <prafullchandr.borch@hcl.com>; Pradnya Ganeshrao Raut <pradnya.ganes@hcl.com>; Prabhu Gorle <prabhu.gorle@hcl.com>; Anjum Firdos . <anjumfirdos.a@hcl.com>; Pothu Shanti <pothu.shanti@hcl.com>; Poornayya Cherukumalla <poornayya.c@hcl.com>; Poonam Khati <poonam.khati@hcl.com>; Poojitha Sai Moganti <poojithasai.moganti@hcl.com>; Pooja Rajput <pooja.rajput@hcl.com>; Pooja Patel <pooja.patel@hcl.com>; Ponaganti Akhila <ponaganti.akhila@hcl.com>; Peruri Naveen <peruri.naveen@hcl.com>; Perumal M <perumal.m@hcl.com>; Ankur Agarwal - ERS, HCL Tech <Ankur.Agarwal@hcl.com>; Pelluru Teja <pelluru.teja@hcl.com>; Pawan Kumar Prajapati <pawankumar.prajapati@hcl.com>; Pavan Kumar Reddy B <pavankumar.reddyb@hcl.com>; Ankur Gupta <ankur-gu@hcl.com>; Ankur Prakash <ankur.prakash@hcl.com>; Pathmaja S <pathmaja.s@hcl.com>; Pasupuleti Ananda Naga Sai Durga Prasad <pasupuleti.anand@hcl.com>; Parvathavarthini Kannan <parvathavarth.kanna@hcl.com>; Partheeban

Durairajan <partheeban.durai@hcl.com>; Panugothu Yashwanth <panugothu.yashwanth@hcl.com>; Pankajsharma R <Pankajsharma.R@hcl.com>; Pandhikunta Ganesh <pandhikunta.ganesh@hcl.com>; Pandhare Rajesh Hanmant <pandharerajesh.hanmant@hcl.com>; Pampadesatish Kumar <pampadesatish.kumar@hcl.com>; Padya Premsingh <padya.premsingh@hcl.com>; Pachimatala Laya <pachimatala.laya@hcl.com>; Oviya P <oviya.p@hcl.com>; Onkar Agnihotri <onkar.agnihotri@hcl.com>; Nitin Sharma <nitin.sh@hcl.com>; Nitin Bhardwaj <nitin.bhardwaj@hcl.com>; Nithin Kumbhamnn <nithin.k@hcl.com>; Nitesh Kumar Mathur <niteshkumar.mathur@hcl.com>; Nishavani N <nishavani.n@hcl.com>; Anurag Singh <singh.anu@hcl.com>; Nisha M Patilkulkarni <nisham.patilkulkarni@hcl.com>; Anurag Reddy Vatticharla <vatticharla.reddy@hcl.com>; Nipun Samarakoon <nipun.samarakoon@hcl.com>; Nikita Vijaysingh Pawar <nikita.vijay@hcl.com>; Nikita . <nikita.n@hcl.com>; Nikhil Gupta <nikhilgup@hcl.com>; Nikhil Gour <nikhil.gour@hcl.com>; Nidhi Kushwaha <nidhi.kushwaha@hcl.com>; Aparna M P <aparna.mp@hcl.com>; Nelabhotla Divya Sree <nelabhotla.divyasree@hcl.com>; Apeksha Mahajan <apeksha.mahajan@hcl.com>; Neeraj Yadav (ECS) <Neeraj.Yadav@hcl.com>; Apoorv Mehrotra <apoorvm@hcl.com>; Nayana R H <nayana.rh@hcl.com>; Narem Harshitha <narem.harshitha@hcl.com>; Nanjula Perera <nanjula.perera@hcl.com>; Aravapalli Swarup <aravapalli.swarup@hcl.com>; Nandish Banakar <nandish.banakar@hcl.com>; Namrata Devanand Mondhe <namratadevane.mondh@hcl.com>; Aravind V <aravind-v@hcl.com>; Nagaraj Vittal Benki <nagarajvittal.benki@hcl.com>; Aravindh Kumar B <aravindh.kumarb@hcl.com>; Nagam Jyostna <nagam.jyostna@hcl.com>; Aravinth L <aravinth.l@hcl.com>; Naga Malleswara Rao Sanka <naga.sanka@hcl.com>; Naga Lakshmi Tadiboina <nagalakshmi.tadib@hcl.com>; Munish Thakur <munish.thakur@hcl.com>; Muniganti Ramesh <muniganti.ramesh@hcl.com>; Munesh Kumar Rawat <muneshkumar.rawat@hcl.com>; Archana Thinetla <archana.thinetla@hcl.com>; Archana. C <archana.c@hcl.com>; Mukkara Hari Krishna <mukkarahari.krishna@hcl.com>; Mubin Chaudhri <mubin.chaudhri@hcl.com>; Vakacharla Mounika <vakacharla.sri@hcl.com>; Monisha. I <Monisha.I@hcl.com>; Mohit Shrivastav <mohit.shrivastav@hcl.com>; Mohd Jeeshan <mohd.jeeshan@hcl.com>; Arolla Anuhyा <arolla.anuhy@hcl.com>; Mohd Fraz . <mohdfraz@hcl.com>; Mohammed Ahsan <mohammed.ahsan@hcl.com>; Ishfaaq Ismath <ishfaaq.ismath@hcl.com>; Mohamed Ahiyar Mohamed <mohamedahiyar.moham@hcl.com>; Metkari Saif Mohammadraphi <metkarisaif.moham@hcl.com>; Meesala Pydi Lokesh Mohan <meesala.pydil@hcl.com>; Md Azharuddin <md.azharuddin@hcl.com>; Maske Tejas Sanjay <masketejas.sanjay@hcl.com>; Marx Marcolenin Torres Martinez <marxmarcolenin.torre@hcl.com>; Marcus Ivan Gonzalez Martinez <marcusivan.gonz@hcl.com>; Mansi Mehto <mansi.mehto@hcl.com>; Arun Kumar Shekhar Deo <arunkumar.shekhardeo@hcl.com>; Manoj Kumar Gupta, (ETS) <ManojKumar.Gupta@hcl.com>; Mani Ram Kumar Siriyala <maniramkumar.siriy@hcl.com>; Manchikanti Ragasree <manchikanti.ragasree@hcl.com>; Manav Vashistha <manav.vashistha@hcl.com>; Mamtha C <maamtha.c@hcl.com>; Mamatha Nunna Gajula <mamathanunna.gajula@hcl.com>; Arunachalam S <arunachalam-s@hcl.com>; Mallu Ganesh Kumar Reddy <malluganeshku.reddy@hcl.com>; Mallela Aravind <mallela.aravind@hcl.com>; Mahesh Mullapudi <mahesh.mullapudi@hcl.com>; Madhuri Chava <madhuri.chava@hcl.com>; Arunpandi D <arunpandi.d@hcl.com>; Madhu Sudhana Rao Perumalla <Madhusudhanarao.P@hcl.com>; Maddipatla Kushwanth <maddipatla.kushwanth@hcl.com>; Macha Yeswanth <macha.yeswanth@hcl.com>; Arvindkumar Aggarwal <arvindkumara@hcl.com>; M Hemalatha <hemalatham@hcl.com>; Lokesh Aathukuri <lokesh.aathukuri@hcl.com>; Lokendra Shekhawat <lokendra.shekhwat@hcl.com>; Lohith N K <lohit.nk@hcl.com>; Asfaq Hussain <asfaq.hussain@hcl.com>; Pagidipalli Asha <pagidipalli.asha@hcl.com>; Linga Amrutha <linga.amrutha@hcl.com>; Likitha P <likitha.p@hcl.com>; Lavanya Singanamala <lavanya.singanamala@hcl.com>; Ashini Rathnапала <ashini.rathnапала@hcl.com>; Lambe Gautam Ashok <lambegautam.ashok@hcl.com>; Kuruva Pavan Kalyan <kuruva.pavankalyan@hcl.com>; Kuppala Susmitha <kuppala.susmitha@hcl.com>; Kundrapu Neelima <kundrapu.neelima@hcl.com>; Chaudhari Kunal Prabhakar <chaudhari.kunal@hcl.com>; Kunal Agrawal <kunal.agrawal@hcl.com>; Kumpati Raju <kumpati.raju@hcl.com>; Ashutosh Mehta <ashutosh.mehta@hcl.com>; Kumbha Sridevi <kumbha.sridevi@hcl.com>; Kumaresha S <kumaresha.s@hcl.com>; Ashwini H <ashwini.h@hcl.com>; Kumar Aishwary <aishwaryk@hcl.com>; Bhosale Kshitija Someshwar <[bosalekshiti.somes@hcl.com](mailto:bhosalekshiti.somes@hcl.com)>; Kshitij Jaiswal <kshitij.jaiswal@hcl.com>; Kritika Sharma <kritis@hcl.com>; Aswitha J <aswitha.j@hcl.com>; Krishnarao Yellanki <krishnarao.yellanki@hcl.com>; Athieeb Mohamed <atheeb.mohamed@hcl.com>; Krishna Teja Garapati <krishnateja.garapati@hcl.com>; Atla Naveen <atla.naveen@hcl.com>; Krishna Kumar Sharma <krishna-kumarsharma@hcl.com>; Koppisetti Pavan Kalyan <koppisetti.pavan@hcl.com>; Kondoju Srinidhi <kondoju.srinidhi@hcl.com>; Konapala Sharmila <konapala.sharmila@hcl.com>; Kolli Venkata Satya Sri Krishna <kollivenkata.saty@hcl.com>; Kolipakula Sai Sri Harsha <kolipakulasai.harsh@hcl.com>; Avadhanam Koushik Sai Santhosh <avadhanam.koush@hcl.com>; Km Anjali Kesharwani <kmanjali.kesharwani@hcl.com>; Km Anjali <km.anjali@hcl.com>; Kishore P <kishore.p@hcl.com>; Kishor Shankara Gunjegav <kishor.shank@hcl.com>; Kiruthika S <kiruthikas@hcl.com>; Kirti Ghonge <kirti.ghonge@hcl.com>; Avula Manikanta <avula.manikanta@hcl.com>; Khushboo Soni <khushboo.soni@hcl.com>; Khan Ayesha Khatoon Nooruzzama <khan.ayesh@hcl.com>; Kaviya J <kaviya.j@hcl.com>; Kavitha S <kavitha.s@hcl.com>; Kavita Pradhan <kavita.pradhan@hcl.com>; Katakan Abhijit <katakan.abhijit@hcl.com>; Kasipata Samarpana Saraiyah <kasipatasamar.sarai@hcl.com>; Karthik Reddy Kanala <karthikreddy.kanala@hcl.com>; Karthik G <gkarthik@hcl.com>; Karthick Raj Selvaraj <karthickraj.selvaraj@hcl.com>; Karna Venkata Hari Babu <karna.venka@hcl.com>; Karka Hemanth <karka.hemanth@hcl.com>; Kapile Anuj Vinayak <kapileanuj.vinayak@hcl.com>; Kannan Kumar T R <kannan.kumarr@hcl.com>; Kandunuri Rakesh <kandunuri.rakesh@hcl.com>; Badhireddy Sudharshan <badhireddy.sudha@hcl.com>; Kandrekula Lavanya <kandrekula.lavanya@hcl.com>; Kanchem Gurunath <kanchem.gurunath@hcl.com>; Santhosh Kumar Balaga <balaga.kumar@hcl.com>; Kamlesh Mishra <kamlesh.mishra@hcl.com>; Kamil Kuriata <kamil.kuriata@hcl.com>; Kamat Shweta Datta <kamatshweta.datta@hcl.com>; Kamarel Malanda <kamarel.malanda@hcl.com>; 'kalpana.golla@hcl.com' <kalpana.golla@hcl.com>; Kalluru Sushma <kalluru.sushma@hcl.com>; Kakde Amita Bhujangrao <kakde.amita@hcl.com>; Kajave Vaibhav Mahdev <kajavevaibhav.mahdev@hcl.com>; K Pargunan <pargunank@hcl.com>; Jyothi Priya Ujjagiri <jyothipriya.ujjagiri@hcl.com>; Judson Sam V <judsonsam.v@hcl.com>; Juan Antonio Melendres Villa <juanantonio.melen@hcl.com>; Balibani Jyothi <balibani.jyothi@hcl.com>; Jijau Mallikarjun Lohar <jiJaumallikar.lohar@hcl.com>; Jelio Hallies Jaganathan <jelio.hj@hcl.com>; Jayesh Bramhankar <jayesh.bramhankar@hcl.com>; Javier Eduardo Serrano Alvarado <javiereduardo.serra@hcl.com>; Jatin Jaykishan Ramnani <jatinjaykishan.ramna@hcl.com>; Bandaru Harini <bandaru.harini@hcl.com>; Bandaru Naveen Kumar <bandaru.naveenkumar@hcl.com>; Janardhana Rao Regani <janardhana.raoregani@hcl.com>; Janani P <janani.hcl.com>; Jamuna D <jamuna.d@hcl.com>; Jammula Divya <jammula.divya@hcl.com>; Jagan A <jagan.a@hcl.com>; Jadhav Popat Bhimrao <jadhav.popatbhimrao@hcl.com>; Bapanapalli Subha Sri <bapanapalli.subhasri@hcl.com>; Jabadarai S <jabadarai.s@hcl.com>; Ishita Raj <ishitarai.i@hcl.com>; Barupatla Venkatesh <barupatla.venkatesh@hcl.com>; Hemanth K <hemanth-k@hcl.com>; Harshit Pant <harshit.pant@hcl.com>; Harshit Mishra <harshit.mishra@hcl.com>; Harsh Saini <harsh.saini@hcl.com>; Batchu Pavani <batchu.pavani@hcl.com>; Haritha Sankari <haritha.s@hcl.com>; Hari Hara Pavan Koushik Tekuru <hariharapavan.koush@hcl.com>; H Prasan Kumar <hprasan.kumar@hcl.com>; Gyanendra Kumar Rout <gyanendrakumar.rout@hcl.com>; Gurajala Dinesh Kumar Reddy <gurajala.dines@hcl.com>; Guduru Dheekshitha <guduru.dheekshitha@hcl.com>; Guduri Naveen <guduri.naveen@hcl.com>; Gudavally Sai Teja <gudavally.saitje@hcl.com>; Gottipati Nikhitha <gottipati.nikh@hcl.com>; Gorla Mounika <gorla.mounika@hcl.com>; Gollalesh Mallikarjun Budihal <gollaleshmall.budih@hcl.com>; Bharat Jingonda Patil <bharatjingonda.patil@hcl.com>; Gnana Priya Nachu <gnanapriya.nachu@hcl.com>; Girija Muppidi - ERS, HCL Tech <Girija.Muppidi@hcl.com>; Giddaluru Lakshmi Kanth Vara Prasad <giddaluru.laksh@hcl.com>; Bharath Namisha Rajendiran <bharath.namis@hcl.com>; Gelli Rakesh <gelli.rakesh@hcl.com>; Bharathi S <bharathis@hcl.com>; Geja B <geja.b@hcl.com>; Gayatri Chilamakuri <gayatri.c@hcl.com>; Donthu Gayathri <donthu.gayathri@hcl.com>; Gautam Kumar <gautam.ku@hcl.com>; Gaurav Kumar <gaurav-kumar4@hcl.com>; Garikapati Bhavyasri <garikapati.bhavyasri@hcl.com>; Garbham Himabindu <garbham.himabindu@hcl.com>; Gandeppalli Naveen <gandeppalli.naveen@hcl.com>; Mate Fanindra Khushal <matefanindra.khushal@hcl.com>; Faissal Kenzi <faissal.kenzi@hcl.com>; Bhavya Polu <bhavya.polu@hcl.com>; Enjamuri Bhuvaneswari <enjamuri.bhuv@hcl.com>; Eliezer David Gutierrez Hinojosa <eliezerdavid.gutie@hcl.com>; Edulla Bhavani <edulla.bhavani@hcl.com>;

Ebin Joseph <ebin_joseph@hcl.com>; Bhumireddy Chaithanya <bhumireddy.chait@hcl.com>; Bhuvaneshwari K <k-bhuvaneshwari@hcl.com>; Eaga Abhilash <eaga.abhilash@hcl.com>; Dorian Ryan <dorian.ryan@hcl.com>; Doggala Lakshmi Bhargavi <doggalalakshmi.bharg@hcl.com>; Divyansh Sharma <divyanshs@hcl.com>; Divya B <b.divya@hcl.com>; Bindhu Rasathy V <bindhurasathy.v@hcl.com>; Dipali Mokasare <mokasare.tryambak@hcl.com>; Dhoke Pooja Ramesh <dhokepooja.ramesh@hcl.com>; DHIVYA C <DHIVYAC@hcl.com>; Dheerendra Singh <dheerendra.singh@hcl.com>; Devi Raja <devi_ra@hcl.com>; Deepu Pandey <deepuranjan.pandey@hcl.com>; Deepika Subramanian <deepika.subramanian@hcl.com>; Deepan Ramu <deepan.ramu@hcl.com>; Deepak Selvam K <deepakselvam.k@hcl.com>; Deepak Kumar <deepak_kumar1@hcl.com>; Deepak Chittora <deepak.chittora@hcl.com>; Bonga Anilkumar Reddy <bongaankumar.reddy@hcl.com>; Debaprasad Dash Samanta <debaprasad.dashes@hcl.com>; Boopalan S A <boopalans.a@hcl.com>; David Lynch <david.lynch@hcl.com>; Dasthari Ramya <dasthari.ramya@hcl.com>; Daluvai Hemalatha <daluvai.hemalatha@hcl.com>; D Bommuraj <d.bommuraj@hcl.com>; Brindha Vasudevan <brindha.vasu@hcl.com>; Chowhan Harisingh <chowhan.harisingh@hcl.com>; Chitimreddy Praveen Kumar <chitimreddy.prave@hcl.com>; Chennarajasekarreddy Kona <kona.ch@hcl.com>; Cheetharala Lakshmi <cheetharala.lakshmi@hcl.com>; Chedulla Sai Tejeswa Reddy <chedulla.saite@hcl.com>; Chathuraruwan Gawarammana <chathuraruwan.gawar@hcl.com>; Calvin Jude Christy Paul <calvinjude.chris@hcl.com>; 'Carlos Adrian Balderas Castillo' <carlosadrian.balde@hcl.com>; Akula Sai <akula.sai@hcl.com>; Cesar Martinez Nunez <cesar.martineznunez@hcl.com>; Chandaka Vijay <chandaka.vijay@hcl.com>; Chamith De Silva <chamith.desilva@hcl.com>; Chaithanya K.C <chaithanya.kc@hcl.com>

Subject: RE: Reg: Day 3 Notes

Classification: **Confidential**

Handson

1. Create a class named Venue with the following member variables / attributes (Default access)

Data Type	Variable Name
String	name
String	city

Create another class called Main and write a main method to test the above class.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the venue name
M. A. Chidambaram Stadium

Enter the city name
Chennai

Venue Details :

Venue Name : M. A. Chidambaram Stadium
City Name : Chennai

2. Create a class named Player with the following member variables / attributes (Default access)

Data Type	Variable Name
String	name
String	country
String	skill

Create another class named Main and write a main method to test the above class.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the player name
MS Dhoni

Enter the country name
India

Enter the skill

All Rounder

Player Details :

Player Name : MS Dhoni
Country Name : India
Skill : All Rounder

3. Create a class named Delivery with the following public member variables / attributes

Data Type	Variable Name
Long	over

Long	ball
Long	runs
String	batsman
String	bowler
String	nonStriker

Include a method in the class named `displayDeliveryDetails()`. In this method, display the details of the delivery in the format shown in the sample output. This method does not accept any arguments and its return type is `void`.

Create another class called `Main` and write a main method to test the above class.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the over

1

Enter the ball

1

Enter the runs

4

Enter the batsman name

MS Dhoni

Enter the bowler name

Dale steyn

Enter the nonStriker name

Suresh Raina

Delivery Details :

Over : 1

Ball : 1

Runs : 4

Batsman : MS Dhoni

Bowler : Dale steyn

NonStriker : Suresh Raina

4. Create a class named `Player` with the following member variables / attributes (default access)

Data Type Variable Name

String name

String country

String skill

Create another class called `Main` and write a main method to get the player details in a string seperated by comma.Use `String.split()` function to display the details.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the player details

MS Dhoni,India,All Rounder

Player Details

Player Name : MS Dhoni

Country Name : India

Skill : All Rounder

5. Create a class named `Venue` with the following private member variables / attributes

String name

String city

Include appropriate getters and setters.

[Naming Convention:

getters : `getName` `getCity...`

setters : `setName` `setCity...`]

Create another class and write a main method to test the above class. In the main method, get the choice from the user and update the corresponding venue details.

Input and Output Format:

Refer sample input and output for formatting specifications.
All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output :

Enter the venue name
Green Park Stadium
Enter the city name
Kanpur
Venue Details
Venue Name : Green Park Stadium
City Name : Kanpur
Verify and Update Venue Details

Menu
1.Update Venue Name
2.Update City Name
3.All informations Correct/Exit

Type 1 or 2 or 3

2

Enter the city name

Chennai

Venue Details
Venue Name : Green Park Stadium
City Name : Chennai
Verify and Update Venue Details

Menu
1.Update Venue Name
2.Update City Name
3.All informations Correct/Exit

Type 1 or 2 or 3

1

Enter the venue name

Chidambaram Stadium

Venue Details

Venue Name : Chidambaram Stadium
City Name : Chennai
Verify and Update Venue Details

Menu
1.Update Venue Name
2.Update City Name
3.All informations Correct/Exit

Type 1 or 2 or 3

3

Venue Details

Venue Name : Chidambaram Stadium
City Name : Chennai

6. Venue Details**[Note :**

Strictly adhere to the object oriented specifications given as a part of the problem statement.

Use the same class names and member variable names.

Follow the naming conventions mentioned for getters / setters.

Create 3 separate classes in 3 separate files.]

Create a class named Venue with the following private member variables / attributes

Data Type Variable Name

String name

String city

Include appropriate getters, setters and constructors.

Naming Convention :

getters --- getName,etc...

setters --- setName,etc...

Include a default constructor.

Include a 2-argument constructor --- the 1st argument corresponds to the name and the 2nd argument corresponds to the city.

Override the `toString()` method to display the venue details in the following format specified in the output.

Create a class named VenueBO and include the following methods

No	Method Name	Method Description
1	void displayVenueDetails(Venue venue)	In this method, display the

details of the venue.

Input and Output Format:

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Note : The statement " Venue Details" in the output is displayed in the method inside the BO class.

Sample Input and Output :

Enter the venue name

M Chidhambaram Stadium

Enter the city name

Chennai

Venue Details

M Chidhambaram Stadium,Chennai

this() constructor

- used to invoke different constructor of the same class, it should be always present only in the first line otherwise it gives compilation error

```
class A {
    A(){
        System.out.println("1");
    }
    A(int a){
        this("hello");
        System.out.println("2");
    }
    A(String s){
        this();
        System.out.println("3");
    }
}
public class Main {
    public static void main(String[] args) {
        //A a1=new A();
        A a2=new A(10); //1 3 2
        //A a3=new A("hello");
    }
}
```

```
class A {
    A(){
        this(5);
        System.out.println("1");
    }
    A(int a){
        this("hello");
        System.out.println("2");
    }
    A(String s){
        System.out.println("3");
    }
}
public class Main {
    public static void main(String[] args) {
        A a1=new A();
        //A a2=new A(10); //3 2 1
        //A a3=new A("hello");
    }
}
```

Polymorphism - one class which takes many forms - 2 types

1. static/compile time polymorphism
2. Dynamic/runtime polymorphism

Static/Compile time polymorphism

- Using overloading
- 2 types - Method overloading and Constructor overloading

Method overloading

- same method name, but different number, order and datatype of argument, different return type but present in same class

```
class OverloadDemo {
    //Method overloading
    void test() {
        System.out.println("No parameter");
    }
    /*void test(int a) {
        System.out.println("a = "+a);
    }*/
    void test(int a,int b) {
        System.out.println("a = "+a+" b= "+b);
    }
    double test(double a) {
        System.out.println("a = "+a);
        return a*a;
    }
}
public class Main {
    public static void main(String[] args) {
        OverloadDemo ob=new OverloadDemo();
        ob.test();
        ob.test(10); //implicit conversion
        ob.test(10,20);
        double r=ob.test(10.34);
        System.out.println(r);
    }
}
```

Constructor overloading

- same constructor name, but different number, order and datatype of argument, but present in same class

```
class Box {
    double width;
    double depth;
    double height;

    double volume() {
        return width*depth*height;
    }

    //Constructor Overloading
    Box(){
        width=-1;
        depth=-1;
        height=-1;
    }
    Box(double width,double depth,double height){
        this.width=width;
        this.depth=depth;
        this.height=height;
    }
    Box(double len){
        width=depth=height=len;
    }
}
public class Main {
    public static void main(String[] args) {
        Box b1=new Box();
        Box b2=new Box(10,20,15);
        Box b3=new Box(7);
        double vol;
        vol=b1.volume();
        System.out.println("Volume is "+vol); //Volume is -1
        vol=b2.volume();
        System.out.println("Volume is "+vol); //Volume is 3000.0
        vol=b3.volume();
    }
}
```

```

        System.out.println("Volume is "+vol); //Volume is 343.0
    }
}

```

Passing an object as an argument

```

void add(int a){}
void add(Test t){}
void add(Sample s){}

class Box {
    double width;
    double depth;
    double height;

    double volume() {
        return width*depth*height;
    }

    //Constructor Overloading
    Box(){
        width=-1;
        depth=-1;
        height=-1;
    }
    Box(double width,double depth,double height){
        this.width=width;
        this.depth=depth;
        this.height=height;
    }
    Box(double len){
        width=depth=height=len;
    }
    Box(Box b){
        width=b.width;
        depth=b.depth;
        height=b.height;
    }
}
public class Main {
    public static void main(String[] args) {
        Box b1=new Box();
        Box b2=new Box(10,20,15);
        Box b3=new Box(7);
        Box b4=new Box(b2);
        double vol;
        vol=b1.volume();
        System.out.println("Volume is "+vol); //Volume is -1
        vol=b2.volume();
        System.out.println("Volume is "+vol); //Volume is 3000.0
        vol=b3.volume();
        System.out.println("Volume is "+vol); //Volume is 343.0
        vol=b4.volume();
        System.out.println("Volume is "+vol); //Volume is 3000.0
    }
}

class Example {
    void add(int a,int b){
        sop(a+b);
    }
    void add(int a,int b, int c){
        sop(a+b+c);
    }
    void add(int a,int b,int c, int d){
        sop(a+b+c+d);
    }
    void add(int a,int b,int c, int d,int e){
        sop(a+b+c+d+e);
    }
}

```

```

}
```

var args

- used to define variable number of argument, 0 to any
- Available from JDK1.5 onwards
- denoted by ...
- If we are passing combination of arg with var arg, then var arg should be always present only as last arg
- One method can have only one var args and that also should be present as last argument

```
void add(int...a){ //0 or any number of int arg
```

```
}
```

```
void add(String...s){} //0 or any number of string arg
```

```
void add(double...d, String s){} //error
```

```
void add(int a, String...s, float...f){} //error
```

```
class Demo {
```

```
    void add(int...a) {
        for(int a1:a)
            System.out.println(a1);
    }
    void add(boolean...b) {
        for(boolean b1:b)
            System.out.println(b1);
    }
    void add(String s, double...d) {
        System.out.println(s);
        for(double d1:d)
            System.out.println(d1);
    }
}
```

```
public class Main {
```

```
    public static void main(String...args) {
        Demo d=new Demo();
        //d.add();
        d.add(2,3,4,5);
        d.add(1,2,3,4,5,6,7,8,9);
        d.add(true, false, true, true, false);
        d.add("hello", 1.2, 3.4, 5.6);
    }
}
```

Access specifiers - public, private, protected, default - can be applied for all (ie) class, interface, method, variables, constants

Non access specifiers - It has its own restrictions (ie) cannot be applied for everything

static keyword

- It is a non access specifier
- Whenever a class is declared as static, there is no need to create an object for that class, outer class cannot be declared as static only inner class can be static

```
class A { //outer class
    class B { //inner class
    }
}
```

- Whenever a method is declared as static, if it is present in same class then we can access using "methodname", if it is present in different class then we can access using "classname.methodname"

- Whenever a variable is declared as static it will acts like global variable, if it is present in same class then we can access using "variablename", if it is present in different class then we can access using "classname.variablename"

- static method can access only static content, if it is not static it is accessed by creating an object

```
class Sample {
    int a;
    public static void main(String[] args){
        System.out.println(a); //compilation error, static method can access only static content
    }
}
```

```
class Sample {
```

```

static int a;
boolean b;
public static void main(String[] args){
    System.out.println(a); //0
    Sample s=new Sample();
    System.out.println(s.b); //false
}
}

```

- When we want to execute any task before main(), then we can go for static block which will be executed before main() method

Syntax:

```

static {
    //stmt
}

```

- static method can override only another static method

```

public class Main {
    static int a=4;
    static int b;

    static void method(int x) {
        System.out.println(x); //10
        System.out.println(a); //4
        System.out.println(b); //8
    }

    static {
        System.out.println("Static block initialized");
        b=a*2;
    }
    public static void main(String...args) {
        method(10);
    }
}

```

```

class Sample {
    static int a=10;
    static int b=15;
    static void add() {
        System.out.println(a);
    }
}
public class Main {
    public static void main(String...args) {
        Sample.add(); //10
        System.out.println(Sample.b); //15
    }
}

```

System.out.println()/System.in

System is a predefined class

out is an object of PrintStream class and it is static variable in System class

println() method of PrintStream class

```

public class System {
    static PrintStream out;
    static InputStream in;
}

```

static import

- Available from JDK1.5 onwards, only for static method and static variables
- Normally we can invoke static method and static variable using classname.methodname and classname.variablename, but if we want to invoke static method and static variable without classname.methodname and classname.variablename then we can use static import

```

public class Main {
    public static void main(String...args) {

```

```

        double d=Math.sqrt(16);
        System.out.println(d);
    }

}

import static java.lang.System.out;
import static java.lang.Math.sqrt;

public class Main {
    public static void main(String...args) {
        double d=sqrt(16);
        out.println(d);
    }
}

```

Nested class and Inner class

- A class inside another class

```

//Nested class
class A {
    class B {
    }
}

```

- Inside class can be either static or non-static

```

class A {
    class B { //inner class - non static inner class
    }
}

```

```

class A {
    static class B { //static class
    }
}

```

- Outer class properties can be accessed inside inner class only if it is instance variable or should be declared as final, otherwise it is an error
- We cant access inner class properties in outer class

```

class Outer{
    int outer=10; //Instance variable

    void test() {
        Inner i=new Inner();
        i.display();
    }
    class Inner { //Inner class
        int y=5;
        void display() {
            System.out.println(y+" "+outer);
        }
    }
}

public class Main {
    public static void main(String...args) {
        Outer o=new Outer();
        o.test();
    }
}

```

```

class Outer{
    int outer=10; //Instance variable

    class Inner { //Inner class
        int y=5;
        void display() {
            System.out.println(y+" "+outer);
        }
    }
}

```

```

public class Main {
    public static void main(String...args) {
        Outer.Inner o=new Outer().new Inner();
        o.display();
    }
}

class Outer{
    static int outer=10; //Instance variable

    static class Inner { //static class
        int y=5;
        void display() {
            System.out.println(y+" "+outer);
        }
    }
}

public class Main {
    public static void main(String...args) {
        Outer.Inner o=new Outer.Inner();
        o.display();
    }
}

```

Inheritance

- accessing the properties of one class into another class if it is not present
- code reusability
- using "extends" keyword
- single, multilevel, hybrid, hierarchical
- No multiple inheritance

```

class Parent { //Parent/Base/Super class
}
class Child extends Parent { //child/derived/sub class
}

```

//Base class - Single inheritance

```

class Box {
    double width;
    double depth;
    double height;

    double volume() {
        return width*depth*height;
    }

    Box() {
        width=-1;
        depth=-1;
        height=-1;
    }
    Box(double w,double d,double h){
        width=w;
        depth=d;
        height=h;
    }
    Box(double len){
        width=depth=height=len;
    }
    Box(Box ob){
        width=ob.width;
        depth=ob.depth;
        height=ob.height;
    }
}
//Derived class
class BoxWeight extends Box {
    double weight;
}

```

```

BoxWeight(){
    width=-1;
    depth=-1;
    height=-1;
    weight=-1;
}
BoxWeight(double w,double d,double h,double we){
    width=w;
    depth=d;
    height=h;
    weight=we;
}
BoxWeight(double len,double w){
    width=depth=height=len;
    weight=w;
}
BoxWeight(BoxWeight ob){
    width=ob.width;
    depth=ob.depth;
    height=ob.height;
    weight=ob.weight;
}

}

public class Main {
    public static void main(String...args) {
        BoxWeight b1=new BoxWeight();
        double vol;
        vol=b1.volume();
        System.out.println("Volume is "+vol); //Volume is -1.0
        System.out.println("Weight is "+b1.weight);

        BoxWeight b2=new BoxWeight(10,20,15,3);
        vol=b2.volume();
        System.out.println("Volume is "+vol); //Volume is 3000.0
        System.out.println("Weight is "+b2.weight); //3

        BoxWeight b3=new BoxWeight(10,20);
        vol=b3.volume();
        System.out.println("Volume is "+vol); //Volume is 1000.0
        System.out.println("Weight is "+b3.weight); //20

        BoxWeight b4=new BoxWeight(b1);
        vol=b4.volume();
        System.out.println("Volume is "+vol); //Volume is -1.0
        System.out.println("Weight is "+b4.weight); //-1
    }
}

```

super keyword

- used to access base class constructor, base class method and base class variable.
- Only in the case accessing base class constructor, super keyword should be present in first line
- super() and this() cant be used simultaneously

```

//Base class - Single inheritance
class Box {
    private double width;
    private double depth;
    private double height;

    double volume() {
        return width*depth*height;
    }

    Box() {
        width=-1;
        depth=-1;
        height=-1;
    }
    Box(double w,double d,double h){
        width=w;

```

```
        depth=d;
        height=h;
    }
    Box(double len){
        width=depth=height=len;
    }
    Box(Box ob){
        width=ob.width;
        depth=ob.depth;
        height=ob.height;
    }
}
//Derived class
class BoxWeight extends Box {
    double weight;

    BoxWeight(){
        super();
        weight=-1;
    }
    BoxWeight(double w,double d,double h,double we){
        super(w,d,h);
        weight=we;
    }
    BoxWeight(double len,double w){
        super(len);
        weight=w;
    }
    BoxWeight(BoxWeight ob){
        super(ob);
        weight=ob.weight;
    }
}

public class Main {
    public static void main(String...args) {
        BoxWeight b1=new BoxWeight();
        double vol;
        vol=b1.volume();
        System.out.println("Volume is "+vol); //Volume is -1.0
        System.out.println("Weight is "+b1.weight); //-1

        BoxWeight b2=new BoxWeight(10,20,15,3);
        vol=b2.volume();
        System.out.println("Volume is "+vol); //Volume is 3000.0
        System.out.println("Weight is "+b2.weight); //3

        BoxWeight b3=new BoxWeight(10,20);
        vol=b3.volume();
        System.out.println("Volume is "+vol); //Volume is 1000.0
        System.out.println("Weight is "+b3.weight); //20

        BoxWeight b4=new BoxWeight(b1);
        vol=b4.volume();
        System.out.println("Volume is "+vol); //Volume is -1.0
        System.out.println("Weight is "+b4.weight); //-1
    }
}

//Base class - Multilevel inheritance
class Box {
    private double width;
    private double depth;
    private double height;

    double volume() {
        return width*depth*height;
    }

    Box() {
        width=-1;
    }
}
```

```

        depth=-1;
        height=-1;
    }
    Box(double w,double d,double h){
        width=w;
        depth=d;
        height=h;
    }
    Box(double len){
        width=depth=height=len;
    }
    Box(Box ob){
        width=ob.width;
        depth=ob.depth;
        height=ob.height;
    }
}
//Derived class
class BoxWeight extends Box {
    double weight;

    BoxWeight(){
        super();
        weight=-1;
    }
    BoxWeight(double w,double d,double h,double we){
        super(w,d,h);
        weight=we;
    }
    BoxWeight(double len,double w){
        super(len);
        weight=w;
    }
    BoxWeight(BoxWeight ob){
        super(ob);
        weight=ob.weight;
    }
}

class Shipment extends BoxWeight {
    double cost;

    Shipment(double w,double d,double h,double we,double c){
        super(w,d,h,we);
        cost=c;
    }
}
public class Main {
    public static void main(String...args) {
        Shipment s1=new Shipment(10,20,15,3,4);
        double vol;
        vol=s1.volume();
        System.out.println("Volume is "+vol); //3000.0
        System.out.println("Weight is "+s1.weight); //3
        System.out.println("Cost is "+s1.cost); //4
    }
}

class A {
    int i;
}
class B extends A {
    int i;

    B(int i,int j){
        this.i=i; //from B class
        super.i=j; //from A class
    }
    void show() {
        System.out.println(i+" "+super.i);
    }
}

```

```

    }
public class Main {
    public static void main(String...args) {
        B b=new B(1,2);
        b.show(); // 1 2
    }
}

```

Scanner class

- used to get input from user based on the datatype
- present in java.util.*
- If we use combination of numbers and String, either u can use dummy line called sc.nextLine() or get all input in string (ie) using nextLine() and later convert to its related datatype

```

int nextInt()
float nextFloat()
double nextDouble()
short nextShort()
byte nextByte()
long nextLong()
boolean nextBoolean()
String nextLine() - read words with space
String next() - read single word without space

```

```

public class Main {
    public static void main(String...args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter age");
        int s2=Integer.parseInt(sc.nextLine());
        //sc.nextLine();
        System.out.println("Enter name");
        String s1=sc.nextLine();

        //Student st=new Student(s1,s2);
        Student st=new Student();
        st.setName(s1);
        st.setAge(s2);
        System.out.println(st.getName()+" "+st.getAge());
    }
}

```

From: Senthil Kumar T

Sent: Friday, August 25, 2023 3:27 PM

To: Yerramsetti V Subrahmanyamaisatyanarayana <yerramsetti.v.subra@hcl.com>; Yerramreddy Sravani <yerramreddy.sravani@hcl.com>; Yellapu Sai Vamsi <yellapu.saivamsi@hcl.com>; Yeleti Vani <yeleti.vani@hcl.com>; Yaramothu Vasantha <yaramothu.vasantha@hcl.com>; Yallamma Arjun <yallamma.arjun@hcl.com>; Aashif Ameer <aashif.ameer@hcl.com>; Y Abhilash Reddy <yabhilash.reddy@hcl.com>; Aastha . <aastha.a@hcl.com>; Viyas Rameshkumar <viyas.rameshkumar@hcl.com>; Vishnu Vardhan S <vishnu.vardhans@hcl.com>; Vishal Kumar <vishalk@hcl.com>; Vipin Nonia <vipin.nonia@hcl.com>; Vinod . <vinod.v@hcl.com>; Vinithraj B <vinithraj.b@hcl.com>; Vinay Kumar Boora <vinaykumar_boora@hcl.com>; Vijayalaxmi Rasankar <vijayalaxmi.rasankar@hcl.com>; Abdul Suhana <abdul_suhana@hcl.com>; Amirisetty Vijay Kumar <amirisettyvij.kumar@hcl.com>; Vighnesh Syam Prakash <vighnesh.syamprakash@hcl.com>; 'Vetrivelmurugan Selvaraj' <vetrivelmurug.selva@hcl.com>; Venugopala Reddy Koppula <venugopalared.koppu@hcl.com>; Venkatesh S <venkates@hcl.com>; Venkata Suresh Alapati <venkatasuresh.alapa@hcl.com>; Venkat Chandu Tulluri <venkatchandu.tulluri@hcl.com>; Vemavarapu Koteswara Rao <vemavarapu.kotes@hcl.com>; Velpula Upendra <velpula.upendra@hcl.com>; Veliginti Haritha <veliginti.haritha@hcl.com>; Abhinav Rajput <abhinav.rajput@hcl.com>; Veeranjaneyulu Gummadidala <veeranjaneyulu_g@hcl.com>; Veerababu Korumilli <veerababu.korumilli@hcl.com>; Veenam Balu Subrahmanyam <veenambala.subra@hcl.com>; Vasamsetti Sumitha <vasamsetti.sumitha@hcl.com>; Vamshi Krishna Bitla <vamshikrishna.bitla@hcl.com>; Abhishek Anand <abhishek.ana@hcl.com>; Valmeti Suresh <valmeti.suresh@hcl.com>; Abhishek Chauhan <abhishekch@hcl.com>; Vaishnavi K <kvaishnavi@hcl.com>; Abhishek Khandelwal <abhishek.khandelwal@hcl.com>; Vaddineni Sankeerthana <vaddineni.sanke@hcl.com>; Utsav Chaubey <utsav.chaubey@hcl.com>; Uthirapathi T <uthirapathi.t@hcl.com>; Usharani Moturi <usharani.m@hcl.com>; Ujjwal Arun Bharadwaj <ujjwalarun.bharadwaj@hcl.com>; Udaraka Gunawardhana <udaraka.gunawardhana@hcl.com>; Twinkle . <