Day 23

Hashtable

- Map is collection of entries where each entry contains key/value pair.
- In Map, we can use any type of instance as a key and value.
- Example:

```
Map<Integer, String> map = new Hashtable<>();
Map<Account, Customer> map = new Hashtable<>();
```

- Consider example:

```
public class Account {
  private int number;
  private String type;
  private float balance;
  public Account() {
    // TODO Auto-generated constructor stub
  }
  public Account(int number, String type, float balance) {
    this.number = number;
    this.type = type;
    this.balance = balance;
  }
  public int getNumber() {
    return number;
  }
  public void setNumber(int number) {
    this.number = number;
  }
  public String getType() {
    return type;
  }
  public void setType(String type) {
    this.type = type;
  }
  public float getBalance() {
    return balance;
  }
  public void setBalance(float balance) {
    this.balance = balance;
  }

  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
```

```
      result = prime * result + number;
      return result;
    }
    @Override
    public boolean equals(Object obj) {
      if (this == obj)
        return true;
      if (obj == null)
        return false;
      if (getClass() != obj.getClass())
        return false;
      Account other = (Account) obj;
      if (number != other.number)
        return false;
      return true;
    }
    @Override
    public String toString() {
      return String.format("%-10d%-15s%8.2f", this.number, this.type,
  this.balance);
    }
  }
```

```
  public class Customer {
    private String name;
    private String email;
    private String contactNumber;
    public Customer() {
      // TODO Auto-generated constructor stub
    }
    public Customer(String name, String email, String contactNumber) {
      this.name = name;
      this.email = email;
      this.contactNumber = contactNumber;
    }
    public String getName() {
      return name;
    }
    public void setName(String name) {
      this.name = name;
    }
    public String getEmail() {
      return email;
    }
    public void setEmail(String email) {
      this.email = email;
    }
    public String getContactNumber() {
      return contactNumber;
    }
    public void setContactNumber(String contactNumber) {
```

```
    this.contactNumber = contactNumber;
  }
  @Override
  public String toString() {
    return String.format("%-15s%-20s%-15s", this.name, this.email,
this.contactNumber);
  }
}
```

```
  import java.util.Map;
  import java.util.Map.Entry;
  import java.util.Set;

  import org.example.domain.Account;
  import org.example.domain.Customer;

  public class MapTest {
    private Map<Account, Customer> map;
    public void setMap(Map<Account, Customer> map) {
      this.map = map;
    }
    public void addEntries(Account[] keys, Customer[] values) {
      if( this.map != null ) {
        for( int index = 0; index < keys.length; ++ index ) {
          Account key = keys[ index ];
          Customer value = values[ index ];
          this.map.put(key, value);
        }
      }
    }
    public Customer findCustomer(int number) {
      if( this.map != null ) {
        Account key = new Account();
        key.setNumber(number);
        if( this.map.containsKey(key)) {
          Customer value = this.map.get(key);
          return value;
        }
      }
      return null;
    }
    public boolean removeEntry(int number) {
      if( this.map != null ) {
        Account key = new Account();
        key.setNumber(number);
        if( this.map.containsKey(key)) {
          this.map.remove(key);
          return true;
        }
      }
      return false;
```

```java
    }
    public void printEntries() {
      if( this.map != null ) {
        Set<Entry<Account, Customer>> entries = this.map.entrySet();
        for (Entry<Account, Customer> entry : entries) {
          Account key = entry.getKey();
          Customer value = entry.getValue();
          System.out.println(key+" "+value);
        }
      }
    }
  }
```

```java
import java.util.Hashtable;
import java.util.Scanner;

import org.example.domain.Account;
import org.example.domain.Customer;
import org.example.test.MapTest;

public class Program {
  private static Scanner sc = new Scanner(System.in);
  private static Account[] getKeys( ) {
    Account[] arr = new Account[ 5 ];
    arr[ 0 ] = new Account( 4627,"Saving", 10000.50f);
    arr[ 1 ] = new Account( 1930,"Current",20000.50f);
    arr[ 2 ] = new Account( 5629,"Loan",30000.50f);
    arr[ 3 ] = new Account( 9356,"Pention",40000.50f);
    arr[ 4 ] = new Account( 7485,"PPF",50000.50f);
    return arr;
  }
  private static Customer[] getValues( ) {
    Customer[] arr = new Customer[ 5 ];
    arr[ 0 ] = new Customer("Chetan","chetan@gmail.com","9527325202");
    arr[ 1 ] = new Customer("Ganesh","ganesh@gmail.com","9527325202");
    arr[ 2 ] = new
Customer("Saishwar","saishwar@gmail.com","9527325202");
    arr[ 3 ] = new Customer("Anup","anup@gmail.com","9527325202");
    arr[ 4 ] = new Customer("Mahesh","mahesh@gmail.com","9527325202");
    return arr;
  }
  public static void acceptRecord( int[] accountNumber ) {
    if( accountNumber != null ) {
      System.out.print("Enter account number    :   ");
      accountNumber[ 0 ] = sc.nextInt();
    }
  }
  private static void printRecord(Customer value) {
    if( value != null )
      System.out.println( value );
    else
```

```
            System.out.println("Account not found");
    }
    private static void printRecord(boolean removedStatus) {
        if( removedStatus )
            System.out.println("Entry is removed.");
        else
            System.out.println("Account not found");
    }
    private static int menuList( ) {
        System.out.println("0.Exit.");
        System.out.println("1.Add Entry.");
        System.out.println("2.Find Customer.");
        System.out.println("3.Remove Entry.");
        System.out.println("4.Print Entries.");
        System.out.print("Enter choice  :   ");
        return sc.nextInt();
    }
    public static void main(String[] args) {
        int choice;
        int[] accountNumber = new int[ 1 ];
        MapTest test = new MapTest();
        test.setMap( new Hashtable<>() );
        while( ( choice = Program.menuList( ) ) != 0 ) {
            switch( choice ) {
            case 1:
                Account[] keys = Program.getKeys();
                Customer[] values = Program.getValues();
                test.addEntries( keys, values );
                break;
            case 2:
                Program.acceptRecord(accountNumber);
                Customer value =  test.findCustomer( accountNumber[ 0 ] );
                Program.printRecord( value );
                break;
            case 3:
                Program.acceptRecord(accountNumber);
                boolean removedStatus =  test.removeEntry( accountNumber[ 0 ]
);
                Program.printRecord( removedStatus );
                break;
            case 4:
                test.printEntries( );
                break;
            }
        }
    }
}
```

HashMap<K,V>

- It is a sub class of AbstractMap<K,V> class and it implements Map<K,V>
- Its implementation is based on Hash table.

- Since it is Map collection, we can not insert duplicate keys but we can insert duplcate value.
- In HashMap<K,V> key and value can be null.
- It is unsynchronized collection. Using Collections.synchronizedMap() method we can consider it synchronized.
- This class is a member of the Java Collections Framework.
- It is introduced in JDK 1.2.
- If we want to use any element of non final type inside Hashtable then non final type should override equals and hashCode method.
- Instantiation:

```
HashMap<Integer, String> map = new HashMap<>( );
Map<Integer, String> map = new HashMap<>( );
```

LinkedHashMap<K,V>

- It is a sub class of HashMap<K,V>.
- Its implementation is based on Hash table and linked list.
- During traversing it maintains order of elements.
- Since it is Map collection, we can not insert duplicate keys but we can insert duplcate value.
- In HashMap<K,V> key and value can be null.
- It is unsynchronized collection. Using Collections.synchronizedMap() method we can consider it synchronized.
- This class is a member of the Java Collections Framework.
- It is introduced in JDK 1.2.
- If we want to use any element of non final type inside Hashtable then non final type should override equals and hashCode method.
- Instantiation:

```
LinkedHashMap<Integer, String> map = new LinkedHashMap<>( );  //OK
HashMap<Integer, String> map = new LinkedHashMap<>( );  //OK
Map<Integer, String> map = new LinkedHashMap<>( );  //OK
```

TreeMap<K,V>

- It is a sub class of AbstractMap<K,V> class and it implements NavigableMap<K,V>
- Its implementation is based on Red Black Tree.
- TreeMap store entries in sorted order.
- Since it is Map collection, we can not insert duplicate keys but we can insert duplcate value.
- In TreeMap<K,V> key can not be null but value can be null.
- It is unsynchronized collection. Using Collections.Collections.synchronizedSortedMap() method we can consider it synchronized.
- This class is a member of the Java Collections Framework.
- It is introduced in JDK 1.2.

- If we want to use any element of non final type inside TreeMap then non final type should implement Comparable interface.
- Instantiation:

  ```
  TreeMap<Integer, String> map = new TreeMap<>( );  //OK
  Map<Integer, String> map = new TreeMap<>( );  //OK
  ```

What is the differece between Collection and Collections?

What is the differece between Collection and Map?

What is the difference between Hashtable and HashMap?

What is the difference between HashMap and LinkedHashMap?

What is the difference between HashMap and TreeMap?

What is the difference between HashSet and Hashtable?

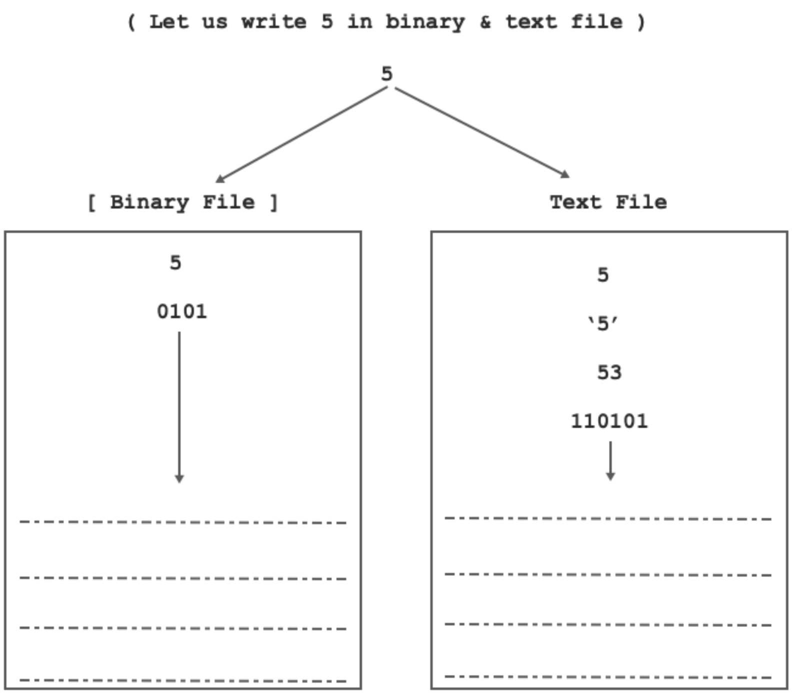
How will you convert HashMap keys and Values into ArrayList?

```
Map<Integer, String> map = new HashMap<>( );
map.put( 1, "DAC");
map.put( 2, "DMC");
map.put( 3, "DESD");

Set<Integet> keys = map.keySet();
List<Integer> list = new ArrayList( keys );

Collection<String> values = map.values();
List<Integer> list = new ArrayList( values );
```

File I/O

- Variable
  - It is temporary conrainter which is used to store record in primary memory(RAM).
- File
  - It is permenant conrainter which is used to store record on HDD.

- Types of files:



[Binary File versus Text File]

  - Text File
    - Examples: .c, .cpp, .java. .cs, .html, css. .js, .txt, .doc, .docs, .xml, .json etc.
    - We can read text file using any text editor.
    - It requires more processing than binary file hence it is slower in performance.
    - If we want to save data in human readable format then we should create text file.
  - Binary File
    - Examples: .jpg, .jpeg, .bmp, .gif, .mp3, .mp4, .obj, .class etc.
    - To read binary file, we must use specific program.
    - It requires less processing than text file hence it is faster in performance.
    - If we dont want to save data in human readable format then we should create binary file.

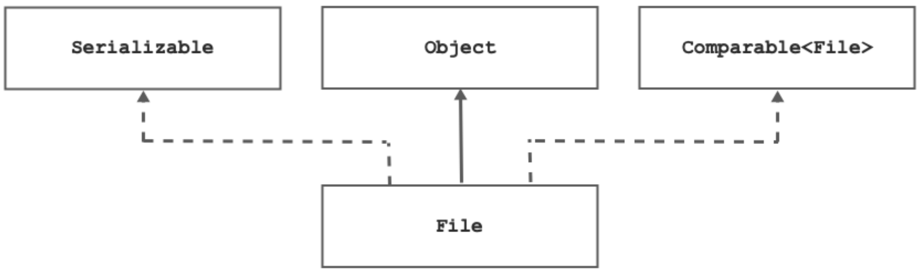
- Stream
  - It is an abtraction(instane) which either consume(read) or produce(write) information from source to destination.
  - Stream is always associated with resource.
  - Standard stream instances of Java programming languages which are associated with Console(Keyboard / Monitor):
    - System.in
    - System.out
    - System.err
- If we want to save data in file the we should use types declared in java.io package.
- java.io.Console class reprsents Console.

```
import java.io.Console;

public class Program {
  public static void main(String[] args) {
    System.out.print("Enter name  : ");
    Console  console = System.console();
    String name = console.readLine();
    //System.out.println("Name  : "+name);
    console.printf("Name  : %s\n", name);
  }
}
```

  - java.io.File class reprsents Physical file on HDD.

File

- It is a class declared in java.io package.



- We can use java.io.File class:
  - To create new file or to remove exisiting file.
  - To create new directory or to remove exisiting directory.
  - To read metadata of file, directory or drive.
- Constructor:
  - public File(String pathname)

    ```
    String pathname = "Sample.txt";
    File file = new File( pathName );
    ```

- Create new file:

```
  public static void main(String[] args) {
  try {
    String pathName = "Sample.txt";
    File file = new File(pathName);
    if( file.exists())
      System.out.println("File already exist.");
    else {
      boolean status = file.createNewFile();
      System.out.println("File creation is successful.");
```

```
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
```

- Remove exisiting file:

```
    public static void main(String[] args) {
    try {
      String pathName = "Sample.txt";
      File file = new File(pathName);
      if( !file.exists())
        System.out.println("File does not exist.");
      else {
        boolean status = file.delete();
        System.out.println("File deletion is successful.");
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

- Create new directory

```
  public static void main(String[] args) {
    try {
      String pathName = "Sample";
      File file = new File(pathName);
      if( file.exists())
        System.out.println("Directory already exist.");
      else {
        boolean status = file.mkdir();
        System.out.println("Directory creation is successful.");
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

- Remove existing directory

```
  public static void main(String[] args) {
    try {
      String pathName = "Sample";
      File file = new File(pathName);
      if( !file.exists())
```

```
          System.out.println("Directory does not exist.");
        else {
          boolean status = file.delete();
          System.out.println("Directory deletion is successful.");
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

- Read Metadata of files:

```
    public static void main(String[] args) {
      try {
        //String pathName = "D:\\Users\\sandeep\\CDAC\\Quiz.txt";
  //Windows
        String pathName = "/Users/sandeep/Desktop/CDAC/Quiz.txt";
  //Linux
        File file = new File(pathName);
        if( file.exists()) {
          System.out.println( "File Name  :   "+file.getName());
  //Quiz.txt
          System.out.println("Parent Directory:   "+file.getParent());
          System.out.println("Length      :   "+file.length());
          System.out.println("Last Modified   :   "+new
  SimpleDateFormat("dd MMM,yyyy hh:mm:ss").format(new
  Date(file.lastModified())));
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
```

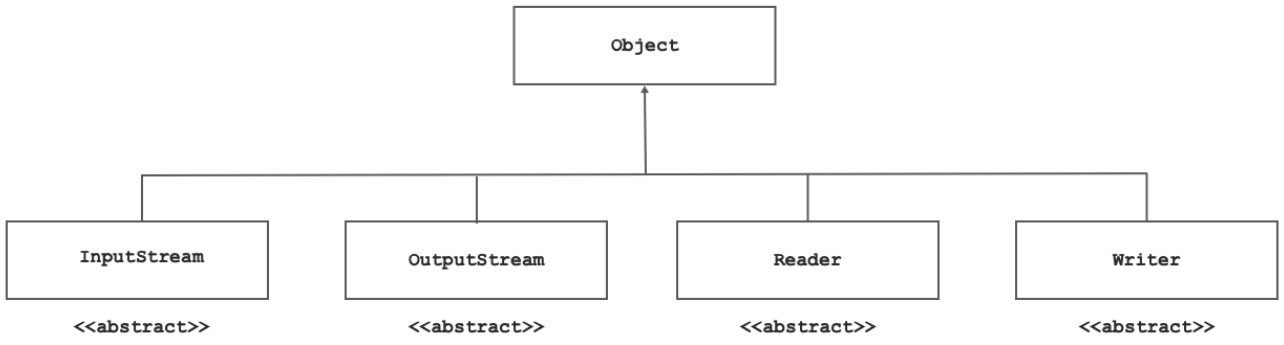
- Read Metadata of directory:

```
    public static void main(String[] args) {
      try {
        String pathName = "/Users/sandeep/Desktop/CDAC/";
        File file = new File(pathName);
        if( file.exists() && file.isDirectory()) {
          //String[] nameList = file.list();
          File[] files = file.listFiles();
          for (File f : files) {
            if( !f.isHidden())
              System.out.println(f.getName());
          }
        }
      } catch (Exception e) {
        e.printStackTrace();
    }
  }
```

Assignment: Write a code in java to simulate windows command prompt?

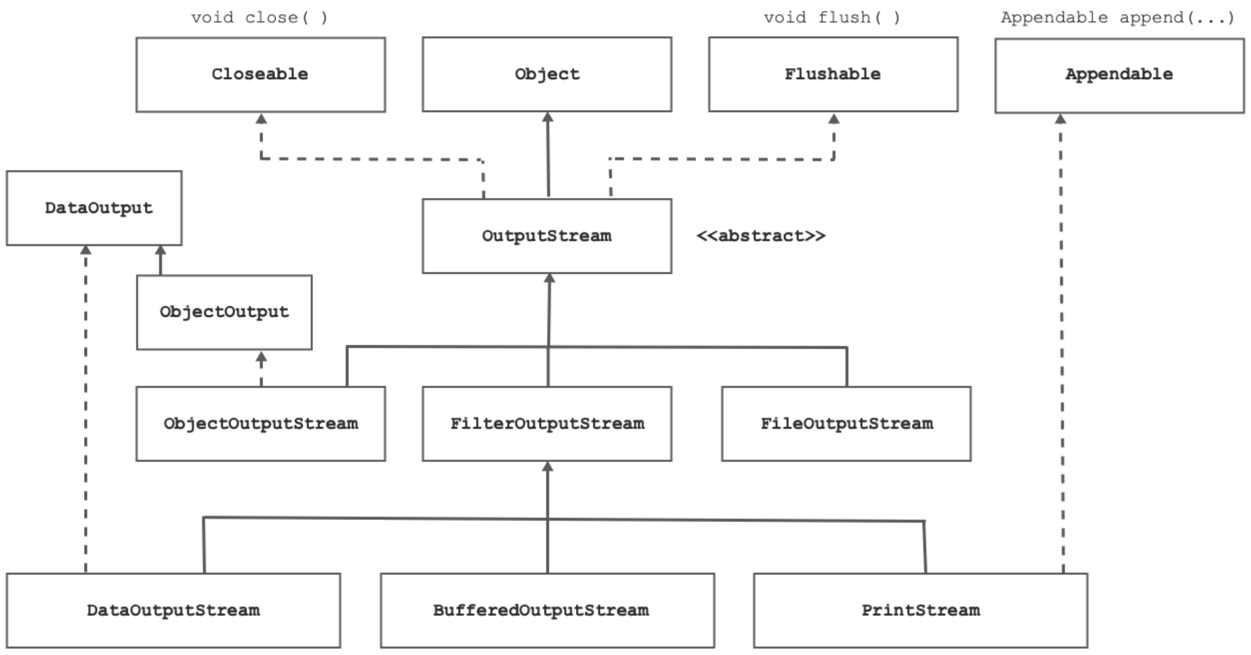
```
Enter path:\>
If it is file then display File metadata
If it is Directory then display File, Directory and Drive metadata
```

File data manipulation

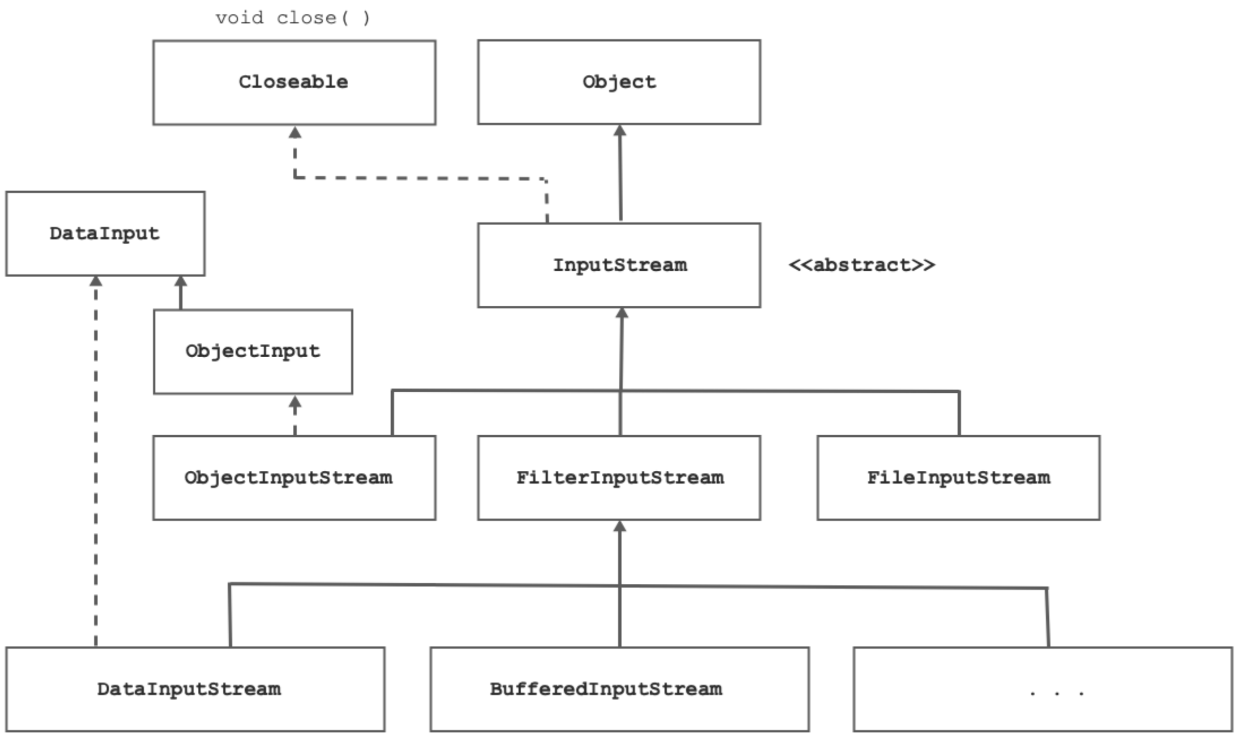


- Interfaces:
    - FileFilter
    - FilenameFilter
    - Closeable
    - Flushable
    - DataInput
    - DataOutput
    - ObjectInput
    - ObjectOutput
    - Serializable
    - Externalizable
- If we want to read/write data into binary file then we should use InputStream, OutputStream and their sub classes.

- Consider OutputStream hierarchy:

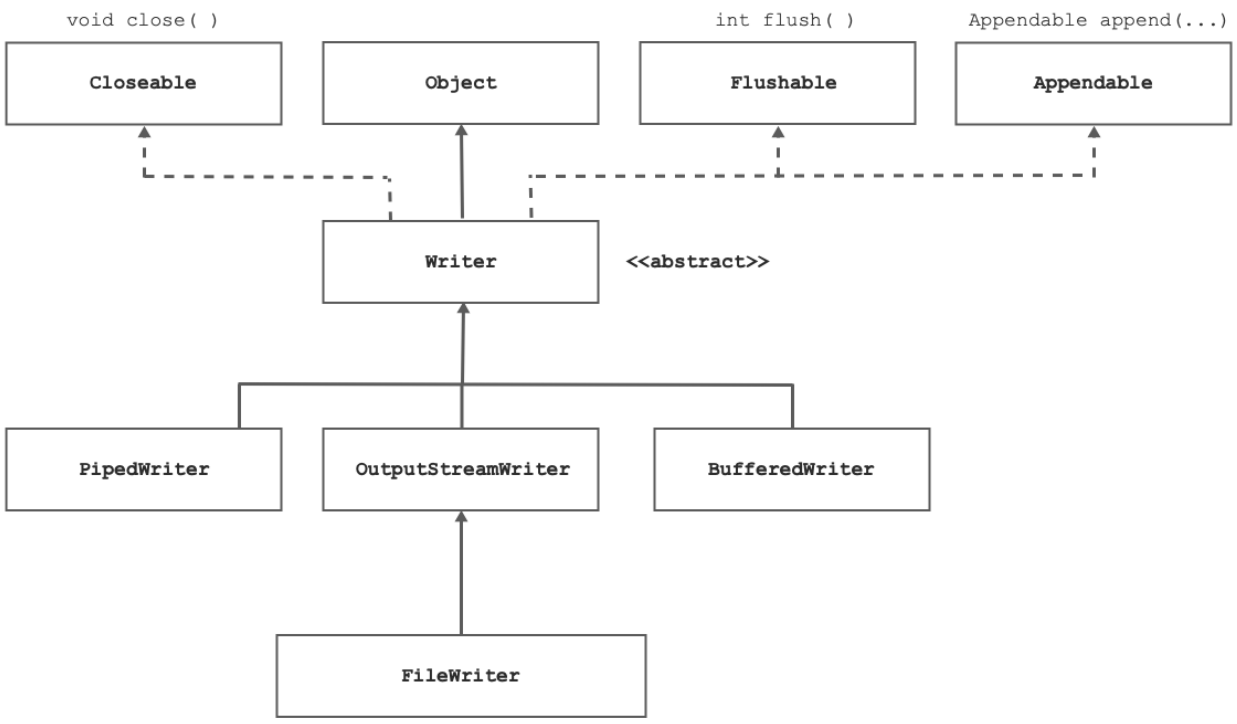


- Consider InputStream hierarchy:

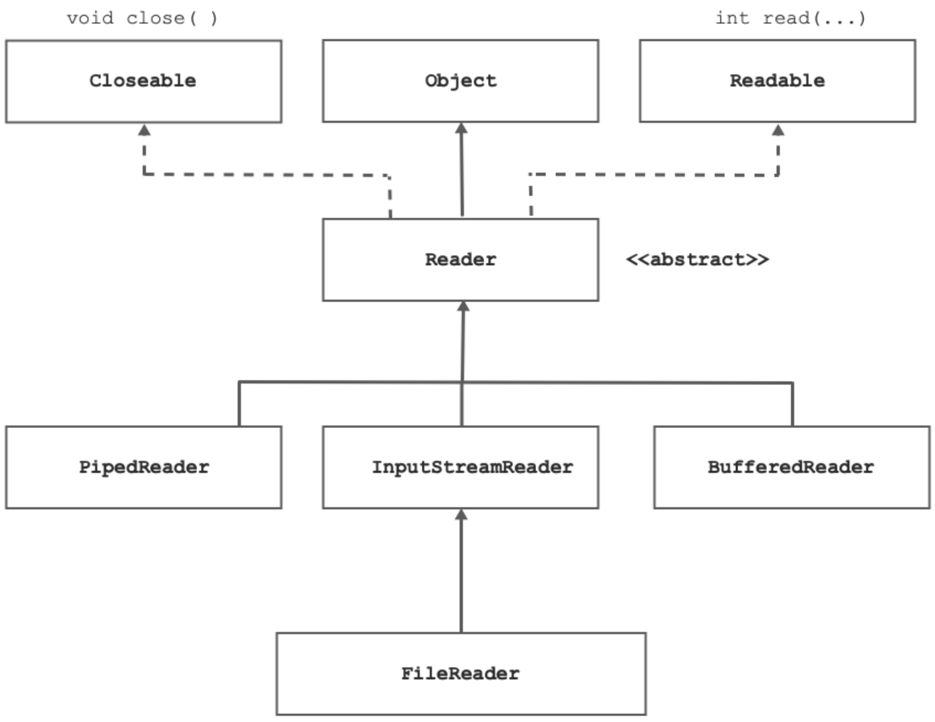


- If we want to read/write data into text file then we should use Reader, Writer and their sub classes.

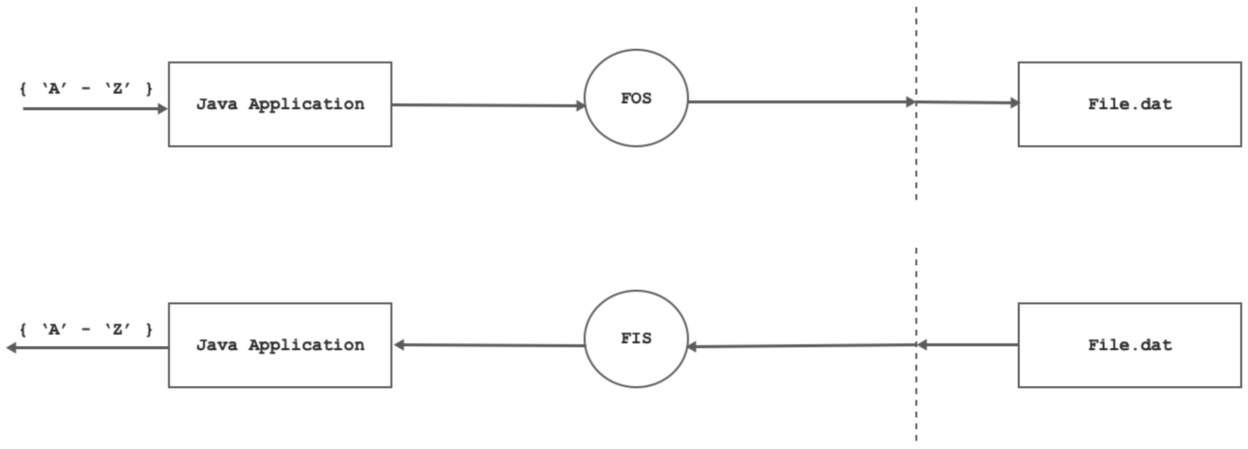
- Consider Writer hierarchy:

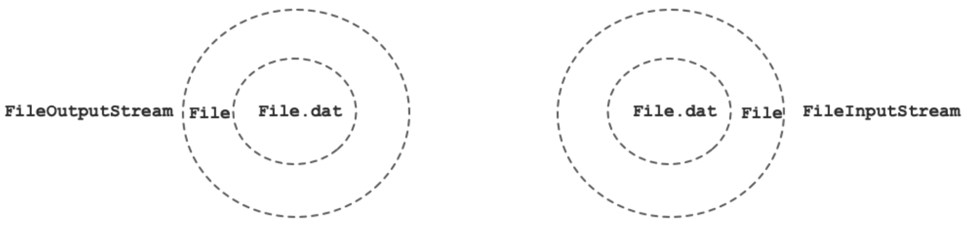


- Consider Reader hierarchy:

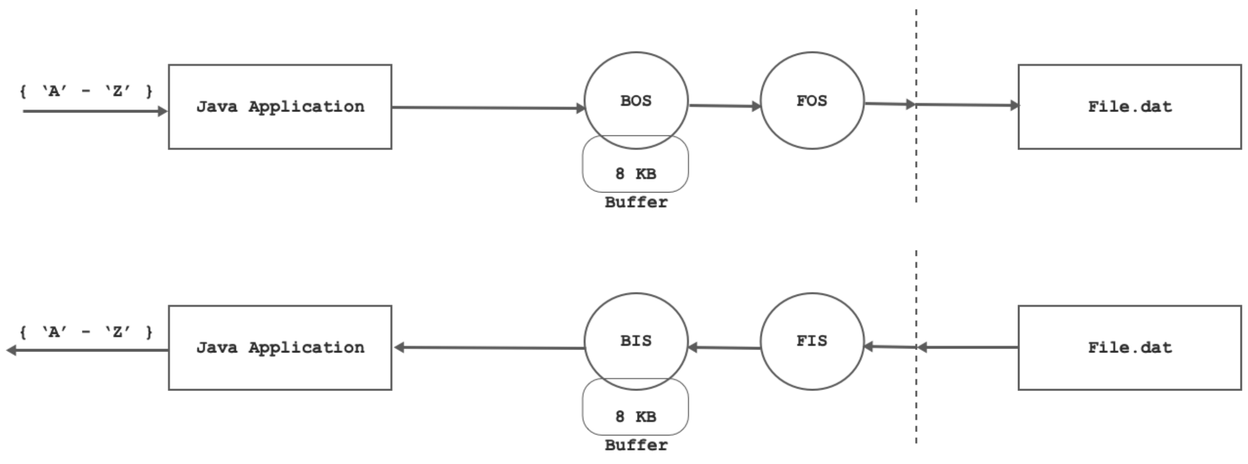


- How to read and write single byte data into file?

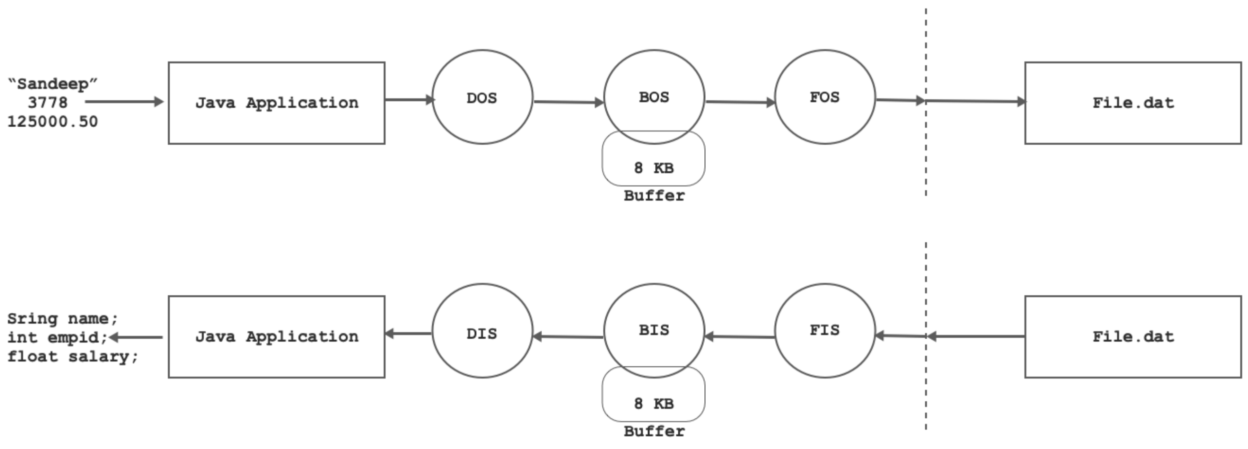




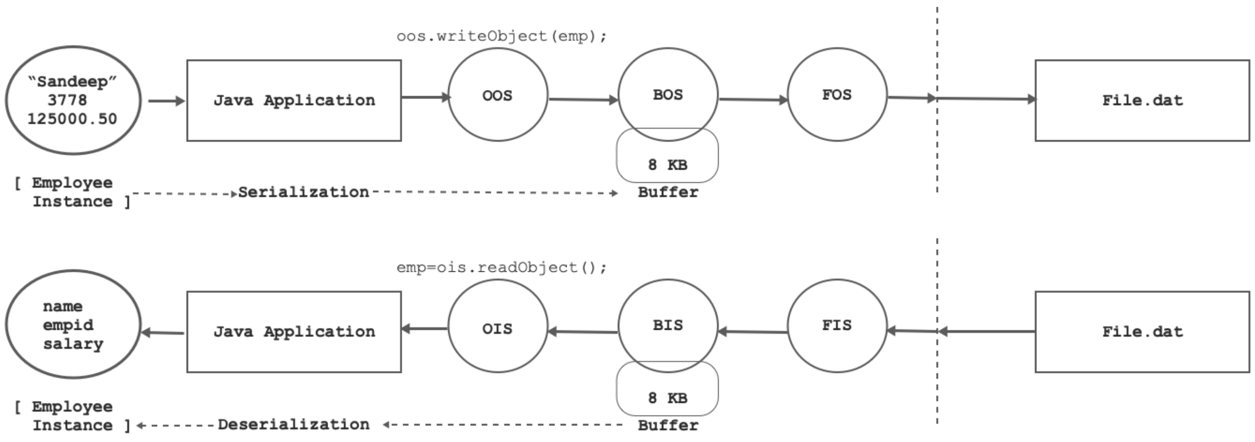
- How to improve performance of read/write operations?



- How to read/write primitive values and Strings to and from files?



- How to serialize / deserialize state of java Instance?



- Process of converting state of Java instance into binary data is called as serialization.
- To serialize Java instance type of instance must implement Serializable interface otherwise we get NotSerializableException.
- If class contains fields of non primitive type then its type must implement Serializable interface. Consider following example:

```
class Date implements Serializable{
}
class Address implements Serializable{
}
class Person implements Serializable{
  private String name;  //Final class. Already implemented
Serializable
  private Address address;
  private Date birthDate;
}
```

- transient is modifier in Java. If we declare any field transient then JVM do not serialize its state.
- Process of converting binary data into Java instance is called as deserialization.