class Dog

package class27;

public class Dog {

private String name;

private String color;

private String breed;

public Dog(String name, String color, String breed) {

setName(name);

setBreed(breed);

setColor(color);

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public String getBreed() {

return breed;

}

public void setBreed(String breed) {

this.breed = breed;

}

@Override

public String toString() {

return "Dog{" +

"name='" + name + '\'' +

", color='" + color + '\'' +

", breed='" + breed + '\'' +

'}';

}

class ArrayListDemo

package class27;

import java.util.ArrayList;

public class ArrayListDemo {

public static void main(String[] args) {

ArrayList<Dog> dogs = new ArrayList<>();

Dog dog1 = new Dog("Tomy", "Black", "German");

Dog dog2 = new Dog("Jony", "White", "BullDog");

Dog dog3 = new Dog("Kimi", "Brown", "German");

dogs.add(dog1);

dogs.add(dog2);

dogs.add(dog3);

/\*dogs.add(new Dog("Tomy", "Black", "German"));

dogs.add(new Dog("Jony", "White", "BullDog"));

dogs.add(new Dog("Kimi", "Brown", "German"));\*

// System.out.println(dogs);

System.out.println(dog1);

System.out.println(dog2);

System.out.println(dog3);}}

public abstract class Insurance

package class27;

public abstract class Insurance {

String insuranceName;

Insurance(String insuranceName){

this.insuranceName=insuranceName;

}

abstract void getQuote();

abstract void cancelInsurance();

@Override

public String toString() {

return "Insurance{" +

"insuranceName='" + insuranceName + '\'' +

'}';}}

class Car extends Insurance{

class Car extends Insurance{

String carModel;

Car(String insuranceName,String carModel) {

super(insuranceName); //calls the parent class constructor

this.carModel=carModel;

}

@Override

void getQuote() {

System.out.println(insuranceName+ "charges 3% of the car value for "+carModel);

}

@Override

void cancelInsurance() {

System.out.println(insuranceName+ " has been canceled");

}

@Override

public String toString() {

return "Car{" +

"carModel='" + carModel + '\'' +

", insuranceName='" + insuranceName + '\'' +

'}';

}

}

class Pet extends Insurance{

String petType;

Pet(String insurance,String petType){

super(insurance);

this.petType=petType;

}

@Override

void getQuote() {

System.out.println(insuranceName+" charges 1500USD for "+petType);

}

@Override

void cancelInsurance() {

System.out.println(insuranceName+" has been canceled for "+petType);

}

@Override

public String toString() {

return "Pet{" +

"petType='" + petType + '\'' +

", insuranceName='" + insuranceName + '\'' +

'}';

}

}

class Health extends Insurance{

Health(String insuranceName) {

super(insuranceName);

}

@Override

void getQuote() {

System.out.println(insuranceName+ "charges 2000USD for health");

}

@Override

void cancelInsurance() {

System.out.println(insuranceName+" has been canceled");

}

@Override

public String toString() {

return super.toString();

}

}

HashMap

package class27;

import java.util.HashMap;

public class MapDemo {

public static void main(String[] args) {

HashMap<String,String> data=new HashMap<>();

data.put("UserName","Anton123"); // This is how we store info inside a map

data.put("Password","pass123");

System.out.println(data);

System.out.println(data.get("Password")); // This is how we can get back the values}}

========================

MapDemo1

package class27;

import java.util.HashMap;

public class MapDemo1 {

public static void main(String[] args) {

HashMap<String,Double> fruit=new HashMap<>();

fruit.put("Apple",1.99); // to Store the values in a map

fruit.put("Mango",2.99);

fruit.put("Orange",4.99);

fruit.put("Banana",10.1);

System.out.println(fruit);

System.out.println(fruit.get("Orange"));

fruit.remove("Mango"); // deletes this key and its value from the map

System.out.println(fruit);

System.out.println(fruit.containsKey("Apple")); // tells us if a key is present inside the map or not

System.out.println(fruit.containsKey("Kiwi"));

System.out.println(fruit.containsValue(4.99)); // searches the map for this value

System.out.println(fruit.isEmpty());

fruit.replace("Apple",2.99);

System.out.println(fruit); }}

MapDemo2

package class27;

import java.util.LinkedHashMap;

public class MapDemo2 {

public static void main(String[] args) {

LinkedHashMap<String,Double> beautyProducts=new LinkedHashMap<>();

beautyProducts.put("Foundation",50.5);

beautyProducts.put("Blush",20.0);

beautyProducts.put("Lipstick",10.5);

LinkedHashMap<String,Double> cosmetics=new LinkedHashMap<>();

cosmetics.put("Soap",10.2);

cosmetics.put("conditioner",20.5);

cosmetics.put("Shampoo",30.99);

LinkedHashMap<String,Double> grocery=new LinkedHashMap<>();

grocery.putAll(beautyProducts);

System.out.println(grocery);

grocery.putAll(cosmetics);

System.out.println(grocery);

// grocery.remove(beautyProducts); Does not work we can't remove all entries from the map

System.out.println(grocery);

grocery.clear();

System.out.println(grocery);}}

MapDemo3

package class27;

import java.util.Collection;

import java.util.HashMap;

public class MapDemo3 {

public static void main(String[] args) {

HashMap<String,Double> fruit=new HashMap<>();

fruit.put("Apple",1.99); // to Store the values in a map

fruit.put("Mango",1.99);

fruit.put("Orange",4.99);

fruit.put("Banana",10.1);

fruit.put("Banana",12.0);

System.out.println(fruit);

System.out.println(fruit.get("Banana"));

Collection<String> keySet=fruit.keySet(); //returns all the keys in the form of Set

System.out.println(keySet);

Collection<Double> values=fruit.values(); // returns all the values in the form of a collection

System.out.println(values);}}

MapDemo4

package class27;

import java.util.HashMap;

public class MapDemo4 {

public static void main(String[] args) {

HashMap<String,Double> fruit=new HashMap<>();

fruit.put("Apple",1.99); // to Store the values in a map

fruit.put("Mango",1.99);

fruit.put("Orange",4.99);

fruit.put("Banana",10.1);

/\* Collection<String> allKeys=fruit.keySet(); // getting all the keys in the form of a set

System.out.println(allKeys);

Iterator<String> iterator= allKeys.iterator(); // Getting an iterator on that set

while (iterator.hasNext()){

String item=iterator.next();

if(item.contains("n")){

iterator.remove();

}

}\*/

fruit.keySet().removeIf(x-> x.contains("n"));

System.out.println(fruit);}}

MapDemo5

package class27;

import java.util.Collection;

import java.util.HashMap;

import java.util.Iterator;

public class MapDemo5 {

public static void main(String[] args) {

HashMap<String,Double> fruit=new HashMap<>();

fruit.put("Apple",1.99); // to Store the values in a map

fruit.put("Mango",1.99);

fruit.put("Orange",4.99);

fruit.put("Banana",10.1);

Collection<Double> values=fruit.values(); // returns all the values from a map

Iterator<Double> iterator=values.iterator();

while (iterator.hasNext()){

Double value= iterator.next();

if(value>2){

iterator.remove();}}

System.out.println(fruit);

// fruit.values().removeIf(x->x>2); same results as code from line 15 to 24

System.out.println(fruit);}}

MapDemo6

package class27;

import java.util.HashMap;

public class MapDemo6 {

public static void main(String[] args) {

HashMap<String,Double> fruit=new HashMap<>();

fruit.put("Apple",1.99); // to Store the values in a map

fruit.put("Mango",1.99);

fruit.put("Orange",4.99);

fruit.put("Banana",10.1);}}

Notes

ArrayList vs LinkedList:

ArrayList and LinkedList both implement the list interface. But internal implementation is different. LinkedList

is better for manipulation(adding new data at run time in-between or in the start of the list or removing

data) of data. ArrayList is better when most of the time we use get method.

Sets:

Sets are another set of datastructures in collections framework that allow us to store only unique data.

as we can convert any data structure in collections framework we can use sets to remove duplicate data from

Lists we can also use sets.

Types of sets:

There are many types of Sets, but we will be discussing only 1) HashSet 2) LinkedHashset 3) TreeSet.

HashSet:

Upside: It is most memory efficient, and it requires less space as compared to other types of sets.

Downside: HashSet does not maintain the insertion order.

LinkedHashSet:

UpSide: order of the elements is maintained.

DownSide: it takes more memory, and it is slower than Hashset.

TreeSet:

UpSide: Data is sorted in natural order.

DownSide: It is slowest, and it takes more memory than Hashset.

Demos of storing the objects of a custom class

and how we can iterate through sets

Important interview question:

What is toString() method?

toString method is used to print instance fields of a class whenever we try to print the object of a class println

method internally calls this method. This method is inherited from the Object class. All the casses that we create

in java automatically inherited from Object class.

Task1

package class27;

import java.util.ArrayList;

public class Task1 {

public static void main(String[] args) {

/\*

Create a class Insurance that will have an attribute as insuranceName and unimplemented behaviour as

getQuote and cancelInsurance. Create 3 subclasses Car, Pet, Health. Car class has it’s own attribute as

carModel and Class Pet has petType attribute. Create 3 objects of the sub classes and store them in ArrayList.

Using for loop/advanced for loop/ iterator access all methods of the class.

\*/

ArrayList<Insurance> insurances=new ArrayList<>();

insurances.add(new Car("Adam","Tesla Model s"));

insurances.add(new Pet("Geico","Dog"));

insurances.add(new Health("State life"));

System.out.println(insurances);

for (Insurance i:insurances

) {

i.getQuote();

i.cancelInsurance();}}}

Task2

package class27;

imprt java.util.LinkedList;

public class Task2 {

/\* Create a Card class that will have interest rate field and card type

and a constructor that will initialize the fields.

Create 3 objects of different card and store them into LinkedList.

Using for loop/advanced for loop/ iterator access all methods of the class.

\*/

public static void main(String[] args) {

LinkedList<Card> cards=new LinkedList<>();

cards.add(new Card(10,"Credit card",2000));

cards.add(new Card(15,"Credit card",25000));

cards.add(new Card(20,"Credit card",100));

for (Card card:cards

) {

card.printBalance();

}

}

}

class Card{

double interestRate;

String cardType;

double balance;

public Card(double interestRate, String cardType, double balance) {

this.interestRate = interestRate;

this.cardType = cardType;

this.balance=balance;

}

void printBalance(){

System.out.println(balance);

}

@Override

public String toString() {

return "Card{" +

"interestRate=" + interestRate +

", cardType='" + cardType + '\'' +

'}';

}

}