Exam-1 Study Guide

These are sample questions for exam-1. The exam will be closed book. But I provide a sheet that provides some information. The sheet is shown at the end of this file. If you want me to add anything else, let me know and I will add it to this sheet.

There will be some multiple-choice questions on the test. I have given you some sample questions so you know the type of multiple choice questions. Study these questions as well. Some of the questions may come directly from this file. The solution to the questions are bolded.

1. Which method can main use to get one of the elements of a bag?
   1. pop()
   2. toString()
   3. **remove()**
   4. getOneItem()
   5. None of the above
2. When using abstraction as a design principle you should focus on:
   1. **what you want to do with the data**
   2. how the data is stored
   3. how the data is manipulated
   4. all of the above
   5. a and b
3. The most efficient approach to dealing with a gap left in an array after removing an entry from a bag is to
   1. **replace the entry being removed with the last entry in the array and replace the last entry with null**
   2. replace the entry being removed with the first entry in the array and replace the first entry with null
   3. shift subsequent entries and replace the duplicate reference to the last entry with null
   4. replace the entry being removed with null
4. The node that is easiest to access in a linked-chain is
   1. **the head node**
   2. the tail node
   3. access time is the same for all nodes
   4. it cannot be determined
5. In the LinkedBag implementation, the *numberOfEntries* field
   1. records the number of entries in the current bag
   2. records the number of nodes in the chain
   3. is set to zero in a new chain
   4. **all of the above**
6. To accommodate entries of any class, the bag methods use \_\_\_\_\_\_ .
   1. An inherited type
   2. **a generic type**
   3. a sub class
   4. interface
7. Which behaviors change the contents of a bag?
   1. clear()
   2. remove()
   3. add()
   4. **all of the above**
8. The operation to add an entry to a stack is called a(n)
   1. put
   2. add
   3. **push**
   4. peek

9- Given the following infix expression, which one of the following is the corresponding postfix expression? w + x \* y / z

* 1. **w x y \* z / +**
  2. w x + y z \* /
  3. w x y q + \* /
  4. none of the above  
     + 1. Using the evaluatePostfix algorithm, evaluate the following postfix expression.   
          7 2 + 4 \* (Note that the first operand is 7 and the second operand is 2).
  5. 15
  6. 13
  7. **36**
  8. 18

**The following are short answer questions:**

11- Suppose that s and t are empty stacks and a, b, c, and d are objects. What do the stacks contain after the following sequence of operations executes?

s.push(a);

                      s.push(b);

                      s.push(c);

                        t.push(d);

                        t.push(s.pop());

                        t.push(s.peek());

                        s.push(t.pop());

                        t.pop();

Stack S: a b b

Stack T: d

1. Show the following expression is balanced using a stack.

{*a* (*b* \* *c*) / [*d* + *e*] / *f* )- *g*}

This is not balanced because when you push the closed delimiters, it does not match with open ones.

1. What is the reason for @Override?

It is a safety feature that you write before a method that is overriding another method that the class is inheriting. If there is no such method in the super class, the compiler will give you an error message.

1. What does the statement this(5) do?

This is used inside a constructor to invoke another constructor with one parameter.

1. A static method header says: public static T <T extends Comparable> sort(T a, T b). What does the part that states <T extends Comparable> do?

T extends Comparable says that T that is accepted for this method must be Comparable. In other words, that class (such as Car) must implement the Comparable interface.

1. Explain what auto-boxing of Java means.

What auto-boxing means is that java will automatically converts a primitive type to an object type when needed. Such as when an int type is passed to a method that requires a reference variable, it is converted to Integer type.

1. Write the statement to create an array of Car object references of size 10.

Car[] list = new Car[10];

Note that this does not create 10 Car objects. It only creates 10 reference variables to hold the address of each Car objects.

1. Using the stack show what is the postfix expression for the following infix expression:

(a + b \* c) / (m – n + p)

|  |  |  |
| --- | --- | --- |
| ( |  | ( |
| a | a | ( |
| + | a | (+ |
| b | ab | (+ |
| \* | ab | (+\* |
| c | abc | (+\* |
| ) | abc\*+ |  |
| / | abc\*+ | / |
| ( | abc\*+ | /( |
| m | abc\*+m | /( |
| - | abc\*+m | /(- |
| n | abc\*+mn | /(- |
| + | abc\*+mn- | /(+ |
| p | abc\*+mn-p |  |
| ) | abc\*+mn-p+ | / |
|  | **abc\*+mn-p+/** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. Write a class called Employee that has a member variable called name of type String and a member variable called age of type integer. The class must be Comparable based on the employee age. The Employee class must have a toString method that returns a String containing the name and age.

public class Employee implements Comparable<Employee>{

@Override

public int compareTo(Employee e)

{

if (age > e.age)

return 1;

else if (age == e.age)

return 0;

else

return -1;

}

public String toString()

{

String str = "";

str = str + "name is " + name + " and age is " + age +"\n";

return str;

}

public Employee(String s, int a)

{

name = s;

age = a;

}

public int getAge()

{

return age;

}

//The following are member variables name and age

private String name;

private int age;

}

1. Write the method equals for the Employee class.

@Override

public boolean equals(Object o)

{

Employee e = (Employee) o;

if (age == e.age && name.equals(e.name))

return true;

else return false;

}

1. Write the Java lines to create a LinkedBag called myCompanyL. Create four Employee objects having different names and ages. Add these employees to myCompanyL bag.

public class Main {

public static void main(String[] args) {

BagInterface<Employee> myCompanyL = new LinkedBag<>();

Employee a1 = new Employee("Dave", 54);

Employee b1 = new Employee("Jack", 68);

Employee c1 = new Employee("Chris", 46);

Employee d1 = new Employee("Ron", 28);

myCompanyL.add(a1);

myCompanyL.add(b1);

myCompanyL.add(c1);

myCompanyL.add(d1);

}

1. Write the Java lines to create an ArrayBag called myCompanyA. Create four other Employee objects having different names and ages. Add these employees to myCompanyA bag.

public class Main {

public static void main(String[] args) {

BagInterface<Employee> myCompanyA = new ArrayBag<>();

Employee a2 = new Employee("Dave", 54);

Employee b2 = new Employee("Jill", 49);

Employee c2 = new Employee("Chris", 46);

Employee d2 = new Employee("Tracy", 33);

myCompanyA.add(a2);

myCompanyA.add(b2);

myCompanyA.add(c2);

myCompanyA.add(d2);

}

1. Write the code in main to copy myCompanyA to a LinkedBag.

BagInterface<Employee> myCompanyAcopyL = new LinkedBag<>();

BagInterface<Employee> myTempCompanyA = new ArrayBag<>();

while (!myCompanyA.isEmpty())

{

temp = myCompanyA.remove();

myCompanyAcopyL.add(temp);

myTempCompanyA.add(temp);

}

while (!myTempCompanyA.isEmpty())

{

temp = myTempCompanyA.remove();

myCompanyA.add(temp);

}

1. Write the code in main to print the name of the oldest person in each company.

int max = 0;

Employee  maxEmployee = null;

if (!myCompanyA.isEmpty())

{

temp = myCompanyA.remove();

myTempCompanyA.add(temp);

max = temp.getAge();

maxEmployee = temp;

}

while (!myCompanyA.isEmpty())

{

temp = myCompanyA.remove();

myTempCompanyA.add(temp);

if (temp.getAge() > max)

{

max = temp.getAge();

maxEmployee = temp;

}

}

while (!myTempCompanyA.isEmpty())

{

temp = myTempCompanyA.remove();

myCompanyA.add(temp);

}

System.out.println("Oldest in CompanyA: ");

System.out.println(maxEmployee);

max = 0;

maxEmployee = null;

if (!myCompanyL.isEmpty())

{

temp = myCompanyL.remove();

myCompanyL2.add(temp);

max = temp.getAge();

maxEmployee = temp;

}

while (!myCompanyL.isEmpty())

{

temp = myCompanyL.remove();

myCompanyL2.add(temp);

if (temp.getAge() > max)

{

max = temp.getAge();

maxEmployee = temp;

}

}

while (!myCompanyL2.isEmpty())

{

temp = myCompanyL2.remove();

myCompanyL.add(temp);

}

System.out.println("Oldest in CompanyL: ");

System.out.println(maxEmployee);

1. Write the code to print all the employees working for both companies.

Employee temp;

BagInterface<Employee> myCompanyL2 = new LinkedBag<>();

System.out.println("BEFORE: \n" + myCompanyL);

while (!myCompanyL.isEmpty())

{

temp = myCompanyL.remove();

myCompanyL2.add(temp);

if (myCompanyA.contains(temp))

System.out.println("The following employee works for both companies: " + temp);

}

while (!myCompanyL2.isEmpty())

{

temp = myCompanyL2.remove();

myCompanyL.add(temp);

}

1. Write the code toString for the LinkedBag class.

public String toString(){

String str = "";

Node current = firstNode;

while(current != null)

{

str = str + current.data + "\n";

current = current.next;

}

return str;

}

1. What does ADT mean? What is an ADT in Java?

ADT means Abstract Data Type and it defines what the data type will do. It does not talk about the implementation or how the things are done.

In Java interface defines the ADT.

1. What is the difference between implements keyword and extends keyword?

The keyword implements is used for interfaces. It tells the compiler that this class will have all the implementation for the headers given in the interface.

The extends is used for inheritance. When a derived class extends a base class, all the member variables and the methods defined in the base class are inherited for the derived class.

Cheat sheet:

private class Node

{

  private T    data; // Entry in bag

  private Node next; // Link to next node

private Node(T dataPortion)

{

this(dataPortion, null);

} // end constructor

private Node(T dataPortion, Node nextNode)

{

data = dataPortion;

next = nextNode;

} // end constructor

} // end Node

This is how to make a reference variable to the beginning of linked list:

Node current = firstNode;

To traverse the link list you can loop until you get to the end of the linke list or current become null.

while (current != null)

/\*\*

   An interface that describes the operations of a bag of objects.

   @author Frank M. Carrano

   @author Timothy M. Henry

   @version 4.1

\*/

public interface BagInterface<T>

{

/\*\* Gets the current number of entries in this bag.

@return  The integer number of entries currently in the bag. \*/

public int getCurrentSize();

/\*\* Sees whether this bag is empty.

@return  True if the bag is empty, or false if not. \*/

public boolean isEmpty();

/\*\* Adds a new entry to this bag.

    @param newEntry  The object to be added as a new entry.

    @return  True if the addition is successful, or false if not. \*/

public boolean add(T newEntry);

/\*\* Removes one unspecified entry from this bag, if possible.

       @return  Either the removed entry, if the removal.

                was successful, or null. \*/

public T remove();

/\*\* Removes one occurrence of a given entry from this bag.

       @param anEntry  The entry to be removed.

       @return  True if the removal was successful, or false if not. \*/

   public boolean remove(T anEntry);

/\*\* Removes all entries from this bag. \*/

public void clear();

/\*\* Counts the number of times a given entry appears in this bag.

@param anEntry  The entry to be counted.

@return  The number of times anEntry appears in the bag. \*/

public int getFrequencyOf(T anEntry);

/\*\* Tests whether this bag contains a given entry.

@param anEntry  The entry to locate.

@return  True if the bag contains anEntry, or false if not. \*/

public boolean contains(T anEntry);

public void display();

/\*\* Retrieves all entries that are in this bag.

@return  A newly allocated array of all the entries in the bag.

                Note: If the bag is empty, the returned array is empty. \*/

} // end BagInterface

If You must write a maximum method inside the Bag class Change T as follows:

public final class ArrayBag<T extends Comparable> implements BagInterface<T>