CS20B: HW 1

Nashir Janmohamed
Fall, 2019

September 12, 2019

1 Questions

1. **Q.** What is the difference between an object and a class? Give two examples.

   A. Classes and objects are tightly coupled, in that an object is the fundamental building block of a more complex program or functionality, while a class defines the structure for an object. Objects are instances of a given type that is defined by the class implementation.

   Example 1: A Person class may be instantiated to create objects Mary and Joe, each of which carry Person-related information and are capable of performing Person-like behavior. A Person may have a $name$, $homeAddress$, and $fitnessLevel$, and the person can $eat()$, $sleep$, or even $exercise()$.

   Example 2: An Engine class may be the superclass of two subclasses JetEngine and CarEngine. Objects of the JetEngine and CarEngine classes are objects of the Engine class, but the JetEngine and CarEngine class are not the same class as the Engine class; instead, they inherit from the Engine class. The Engine class provides a blueprint from which to create the JetEngine and CarEngine classes.

2. **Q.** Explain the difference between

   ```
   (1) int a = 3;
   ```

   and

   ```
   (2) Integer a = new Integer(3);
   ```

   A. Code snippet (1) assigns 3 to an *int* variable, a primitive type in Java that is stored on the stack. Code snippet (2) assigns 3 to an *Integer* object and dynamically allocates memory for the object on the heap. *Integer* is a wrapper class for the primitive *int* type that provides useful class methods (i.e. byteValue() which returns the value of the int object as a byte), static methods (such as parseInt(String s) which parses the String argument as a signed decimal integer), fields (i.e. MAX_VALUE and MIN_VALUE allowed for Integer objects and int primitives), and constructors (i.e. from a primitive int value or from a String).

3. For the following concepts/classes define/write at least one method:

   (a) Animal

   A.

   ```
   boolean isHungry() {
      return this.hungry;
   }
   ```

   (b) Dog

   A.

   ```
   void bark() {
      System.out.println("Woof!");
   }
   ```

   (c) User

   A.

   ```
   void logOn(String password) {
      if (this.loggedOn) {
         System.out.println(this.firstName + " " + this.lastName + "is
             already logged on...");
         return;
      }
      if (password.equals(this.password)) {
         System.out.println("Welcome, " + this.firstName + " " +
             this.lastName + "!");
         this.loggedOn = true;
      } else {
         System.out.println("Sorry, that is not the correct password for "
             + this.firstName + " " + this.lastName + "...");
      }
   }
   ```

   (d) File

   A.

   ```
   String getPath() {
      return this.path;
   ```

```
}
```

(e) Database

A.

```java
Data getEntry(int i, int j) {
   if (i >= this.numRows || i < 0) {
      System.out.println("Please enter a row index greater than or
          equal to zero and less than the number of rows (" + numRows +
          ").");
      return null;
   }
   if (j >= this.numCols || j < 0) {
      System.out.println("Please enter a column index greater than or
          equal to zero and less than the number of columns (" + numCols
          + ").");
      return null;
   }
   return this.table[i][j];
}
```

4. **Q.** For the following concepts/classes write at least one instance variable/attribute and an associated constructor you can define taking **more than one input argument/-parameter**:

   (a) MobilePhone

   A.

```java
public class MobilePhone {
   private String brand;
   private String model;
   private String phoneNumber;
   private boolean smartPhone;

   public MobilePhone(String brand, String model, String phoneNumber,
       boolean smartPhone) {
      this.brand = brand;
      this.model = model;
      this.phoneNumber = phoneNumber;
      this.smartPhone = smartPhone;
   }
}
```

   (b) User

   A.

```
public class User {
   protected String firstName;
   protected String lastName;
   protected String username;
   protected String password;
   protected String email;
   protected static Map<String, int> userMap = new HashMap<String,
       int>();

   public User(String firstName, String lastName, String username,
       String password) {
      this.firstName = firstName;
      this.lastName = lastName;
      // check the username, add a suffix (i.e. username + 27
      // == username27) if it is taken/already in the map
      boolean validUsername = true;
      if (User.userMap.containsKey(username)) {
         int suffix = User.userMap.get(username);
         username.concat(Integer.toString(suffix));
         validUserName = false;
         User.userMap.replace(username, suffix, suffix + 1);
      } else {
         User.userMap.put(username, 0);
      }

      this.username = username;

      boolean validPassword = true;
      if (password.length() < 6 || password.equals(username)) {
         this.password = User.generateStrongPassword();
         validPassword = false;
      }
      StringBuilder s = new StringBuilder("Hello " + this.firstName + "
          " + this.lastName + "! Thank you for registering as a user.");
      if (!validUsername || !validPassword) {
         s.append(" We are sending this message because you had an
             invalid ");
         if (!validUsername && !validPassword) {
            s.append("username and password. We have updated them
                accordingly: username = " + this.username
                  + ", password = " + this.password + ".");
         } else if (!validUsername) {
            s.append("username. We have updated it accordingly: username
                = " + this.username + ".");
         } else if (!validPassword) {
            s.append("password. We have updated it accordingly: password
                = " + this.password + ".");
            }
            s.append(" Please respond to this email if you have any
                questions or concerns.");
            MessageUtils.sendEmail(this.email, s.toString());
        }
    }
}
```

(c) File

A.

```
public class File {
    protected String fileType;
    protected long size;

    // size is in bytes
    public File(String fileType, long size) {
        this.fileType = fileType;
        this.size = size;
    }
}
```

(d) Database

A.

```
public class Database {
    protected int numRows;
    protected int numCols;
    protected Data[][] table;

    public Database (int numRows, int numCols) {
        this.numRows = numRows;
        this.numCols = numCols;
        this.table = new Data[numRows][numCols];
    }
}
```

(e) Webpage

A.

```
public class Website {
    protected String domain;
    public Website (String domain) {this.domain = domain;}
}
```

5. **Q.** What is the output of the following program:

```
class T {
   private int t = 20;
   T(){
      t = 40;
   }
}
class Main {
   public static void main (String args []) {
      T t1 = new T();
      System.out.println(t1.t);
   }
}
```

   A. *t* is a private variable in the *T* class, and so trying to access it directly will cause a compilation error. If t were public, or there was a getter function for *t*, then the value of *t* would outputted, 40 in this case.

6. **Q.** Is there any compiler error in the below Java program? Please explain your answer. I don't accept only yes or no.

```
class Point {
   int m_x, m_y;
   public Point (int x, int y) {m_x = x; m_y = y;}
   public static void main (String args []) {
      Point p = new Point();
   }
}
```

   A. The compiler error in the above example is that the main function attempts to create a Point object with a no-argument constructor, when there is not a no-argument constructor defined. If there were no constructors defined, this would not be a problem, as Java would automatically call the default constructor that takes no arguments. Since there is a constructor defined that takes two arguments, the no-argument constructor cannot be called without explicitly defining one.

7. **Q.** Some aspects of each of the following can be modeled with a graph (network) structure. Describe, in each case what the nodes would represent and what the edges would represent. In general, nodes represent concepts/entities, and edges explain what the relationship is between the entities. For example, for navigation, nodes would represent locations (cities, towns, ...) and edges between cities represent if a road exists between them (a weight can be added to denote the distance).

   (a) Airline

      A.

Nodes could represent cities or towns in an airline's network of operation.
Edges could represent whether or not the airline has a flight from one node to another. In this case, the edges may be directed.

(b) Social networks such as Facebook

A.

Nodes could represent users of the social network.
Edges could represent whether or not there is a connection between two users or pages. In the case of a service like Facebook, the edges would be undirected, since a friendship requires two users, while edges on a service like Instagram would be directed, since one person can follow another without being followed back.

(c) The Internet

A.

Nodes could represent distinct webpages on the internet.
Edges could represent whether or not there is a hyperlink leading to a page from another. In this case the edges would be directed, since one page may contain a hyperlink to another regardless of whether the other page has a hyperlink to it.

(d) Movie industry

A.

Nodes could represent individuals who work in the industry (i.e. actors, directors, writers, camera operators, etc...).
Edges could represent whether or not two individuals have worked on a film together. In this case, the edges would be undirected, since one working with another implies a two-way relationship.

(e) Avengers (the movie/comic)

A.

Nodes could represent characters within the Marvel universe.
Edges could represent whether or not two characters within the universe have met.

(f) Amazon.com recommendation system (where a system recommends to you what to buy)

A.

Nodes could represent items in the catalog.
Edges could represent whether a user bought two items in one purchase (where the weight of the edge could be the number of users that bought the two items). The edge(s) with the highest value would be recommended to the user.

8. **Q.** Describe and specify the order of growth of each of the following code sections, using big-O notation (explain your answer!):

   (a)
   ```
   int count = 0;
   for (int i = 1; i < = N; i++)
      count ++;
   ```

   A. The for loop runs N times, and the loop is the only part of the code that relies on the input size N. That means the time complexity is $O(N)$. The use of extra space does not scale with the input, and so this has a space complexity of $O(1)$.

   (b)
   ```
   int count = 0;
   for (int i = 1; i <= N; i++)
      for ( int j = 1; j <= N; j++)
         count ++;
   ```

   A. The outer for loop runs N times, and the inner for loop runs N times per iteration of the outer for loop. This means that in total, the increment of count is run N^2 times. That means the time complexity is $O(N^2)$. The use of extra space does not scale with the input, and so this has a space complexity of $O(1)$.

   (c)
   ```
   int count = 0;
   for (int i = 1; i <= N; i++)
      count ++;
   for (int j = 1; j <= N; j++)
      count ++;
   ```

   A. Even though there are two for loops, they are not nested, and so they run sequentially. First, the first loop runs N times, and then the second loop runs N times. In total, there are $2 * N$ increments of count, which means this is $O(N)$. The use of extra space does not scale with the input, and so this has a space complexity of $O(1)$.

   (d)
   ```
   int count = 0;
   for (int i = 1; i <= N/2; i++)
      for ( int i = 1; i <= N/2; i++)
         count ++;
   ```

   A. The outer for loop runs N/2 times, and the inner for loop runs N/2 times per iteration of the outer for loop. This means that the increment of count is run

$N^2/4$ times. This is just $O(N^2)$. The use of extra space does not scale with the input, and so this has a space complexity of $O(1)$.

(e)
```
public static String reverse ( String s) {
   int n = s.length();
   char [] a = new char[n];
   for (int i = 0; i < n; i++)
      a[i] = s.charAt(n-i-1);
   String reverse = new String (a);
   return reverse;
}
```

A. This is $O(N)$. The for loop runs N times, and constructing the new String 'reverse' takes N time and so the total is $2 * N$, which is $O(N)$.

9. Describe the order of growth of each of the following functions using O notation:

(a) $N^2 + 3 * N$

A. As N becomes very large, $N^2 >> N$, and so this is $O(N^2)$.

(b) $3 * N + N$

A. When simplified, this is $4 * N$, which is just $O(N)$.

(c) $N * (N - 1) + 2$

A. When simplified, $N * (N - 1) + 2 = N^2 - N + 2$. As N becomes very large, $N^2 >> N$ and $N^2 >> 2$, so this is $O(N^2)$.