Unit 9 Practice

Name: Key 3/5/24

Refer to the Frog class to do problem 1.

```
class Frog
{
   private int location;

   public Frog()
   {
      location=0;
   }

   public void hop(int n)
   {
      location+=n;
   }

   public int getLocation()
   {
      return location;
   }

   public String toString()
   {
      String temp = "";
      for(int i = 0;i<location;i++)
         temp+=".";
      temp+="("+location+")";
      return temp;
   }
}
```

1. Write an OddFrog class which extends Frog. An OddFrog's location can only ever be zero or an odd number. After hopping, an OddFrog checks if the location is even, and if it is, it adds 1 to location.

```
class OddFrog extends Frog
{
   public void hop(int n)
   {
      super.hop(n);
      if(getLocation()%2==0)
         super.hop(1);
   }
}
```

2. Write a BFrog class which extends Frog. A BFrog starts at location 100 and moves backwards instead of forwards. See sample code and output here, then write the complete BFrog class.

`BFrog bff = new BFrog();` `System.out.println(bff.getLocation());`	prints 100
`bff.hop(20);` `System.out.println(bff.getLocation());`	prints 80

```
class BFrog extends Frog
{
   public BFrog()
   {
      super.hop(100);
   }
   public void hop(int n)
   {
      super.hop(-1*n);
   }
}
```

(continued on back)

3. Write code to create a regular Frog named **freddy** and then hop it 10 spaces.

```
Frog freddy = new Frog();   freddy.hop(10);
```

4. Write code to create an OddFrog named **frankie** and hop it 11 spaces.

```
OddFrog frankie = new OddFrog();   frankie.hop(11);
```

5. Write code to create a BFrog named **bestie** and hop it 12 spaces.

```
BFrog bestie = new BFrog();   bestie.hop(12);
```

6. Assuming other code has already imported java.util.ArrayList, write one line of code to create an ArrayList named **pond** that could hold all three Frog family objects that you just created.

```
ArrayList<Frog> pond = new ArrayList<Frog>();
```

7. Write three lines of code to add freddy and frankie and bestie to the ArrayList you created in problem 6.

```
pond.add(freddy);
pond.add(frankie);
pond.add(bestie);
```

8. Given the following Book class, write a .equals() method for the Book class that that checks for a null object and then checks all internal values appropriately.

```
public class Book
{
   private String title;
   private String author;
   private int year;

   public Book(String t, String a, int y)
   {
      title = t;
      author = a;
      year = y;
   }
}
```

```
public boolean equals(Object obj)
   {
      if(obj == null) return false;
      Book other = (Book) obj;
      if(year == other.year && title.equals(other.title)
            && author.equals(other.author))
         return true;
      return false;
   }
```