Q1. Study the following program:

```
class A {  public void reply() {System.out.println("Hi!");}  }

class B {  public void reply() {System.out.println("Hey!");}  }

class C {
  public void talkTo(A x) {System.out.println("Hi A object!"); x.reply();}

  public void talkTo(B x) {System.out.println("Hi B object!"); x.reply();}
}

public class Main {

  public static void main(String[] args) {
    A a = new A();
    B b = new B();
    C c = new C();
    c.talkTo(a);
    c.talkTo(b);
  }
}
```

Output:

(a) What is the output from the program?

(b) Draw all variables of `main()`, as well as the objects created. (To make it simple, let's ignore *String[] args*) Also draw the parameters x and their values when `talkTo(A x)` and `talkTo(B x)` run.

(c) Understand what is *method overloading* and illustrate it using this program.

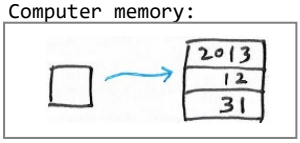
Q2. Consider `Day d = new Day(2013 12 31)`**.**

We are to properly express the relationship between variable d and the created Day object.

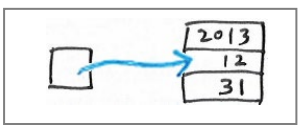
Which one on the right is correct?

Computer memory:

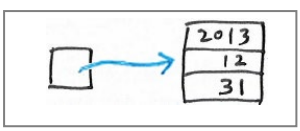
A



B



C



Q3. The code on the right can show the next day of a given day.

Question: Is any part of the code redundant ? (ie. irrelevant and should be removed)

```
..//get values for int y, m, d from user
Day d1 = new Day(y,m,d);
Day d2 = new Day();
d2 = d1.next(); //return the next day of d1
System.out.println("Next of " + d1 + " is " + d2);
```

Q4. [Topic02 - Page 10] Fill in the blanks to explain what happens.

```
Scanner in = new Scanner(System.in);
String s1,s2;
s1 = in.next(); //type "   Today    is a good day."
s2 = in.nextLine();
System.out.println(s1); //"Today"
System.out.println(s2); //"    is a good day“
in.close();
```

```
   Today    is a good day.
Today
    is a good day.
```

i. When `s1 = in.next()` runs, the program **pauses** for input.

ii. After we type " Today is a good day.<enter>", the program wakes up and _____to finish `s1 = in.next()`.

iii. `s1 = in.next()` skips any _____ and then only reads one word (ie. up to _____).
Therefore `s1` gets "Today".

iv. "<space><space><space><space>is a good day.<enter>" is left over in the_____.

v. Then `s2 = in.nextLine()` runs, it doesn't pause for user input but simply gets the _____in the input stream (until <enter>; and <enter> is _____.). Therefore `s2` gets "_____is a good day."

Choose words from below:
pauses
continues
leading spaces
white-space (space,tab,'\n')
input stream
remaining contents
discarded (dropped)
<space><space><space><space>

Q5 [Topic02 - Page 10] Complete the program below according to the sample rundown:

```
What is your name? Peter PAN
How old are you? 28
Hello, Peter PAN.  Next year, you will be 29
```

```
public static void main(String[] args)
{
    String name;
    int age;



    System.out.print("What is your name? ");


    System.out.print("How old are you? ");


    System.out.println("Hello, " + _______ + ".  Next year, you will be " + ___________);
}
```

Q6. [Topic03- Page 1] Complete the blanks and explain what happens to the variable d1.

```
// advance the current day object by 1 day
public void advance()
{
   if (isEndOfAMonth())
   {
       if (month==12)
       {
           year=_____________;
           month=_____________;
           day=_____________;
       }
       else
       {

       }
   }
   else
   {

   }
}
```

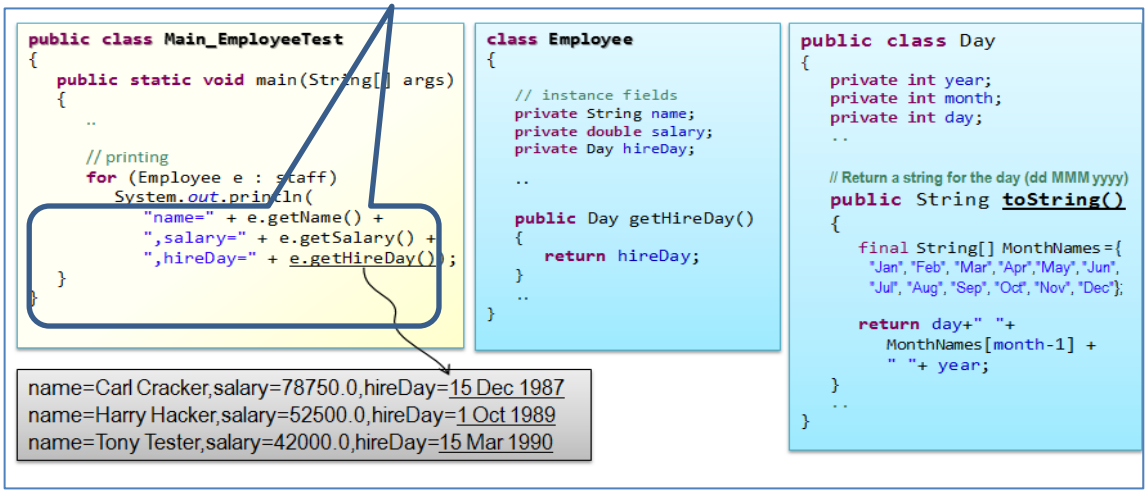
Add this method

```
public class Day
{

  private int year;
  private int month;
  private int day;

  //Constructor
  public Day(int y, int m, int d)
  {
    this.year=y;
    this.month=m;
    this.day=d;
  }

  ..
  // create and return the "next day" of
  // the current day object
  public Day next()
  {
    if (isEndOfAMonth())
      if (month==12)
        return new Day(year+1,1,1);
      else
        return new Day(year,month+1,1);
    else
      return new Day(year,month,day+1);
  }
}
```

Note: some methods return a result, some do not.

```
Day d1 = new Day(2014, 1, 28);
System.out.println(d1.toString());  //Show 28 Jan 2014
________.advance();                 //Advance one day
________.next();                    //Advance one day
System.out.println(d1.toString());  //Show 30 Jan 2014
```

Q7. [Topic03 - Page 3] Explain the output statement in detail steps.



Q8. [Topic03- Page 5] Add the following constructors

```
public class Day {

  private int year, month, day;

  public Day(int y, int m, int d) {
    this.year=y; this.month=m; this.day=d;
  }

  public Day(int y) {



  }

  public Day(int y, int nth_dayInYear) {









  }
```

```
public static void main(String[] args) {
  Day d1;

  d1=new Day(2014,1,28);
  System.out.println(d1); //28 Jan 2014

  d1=new Day(2018);  //The first day in 2018
  System.out.println(d1);  //1 Jan 2018

  d1=new Day(2014,45);  //The 45th day in 2014
  System.out.println(d1);  //14 Feb 2014
}
```

Rundown:

28 Jan 2014
1 Jan 2018
14 Feb 2014

Q9. [Topic03- Page 6] Complete the blanks in the extended program:

- **An object variable:**
  - ☆ Doesn't actually contain an object.
  - ☆ It is used to refer to an existing object of matching type.

```
public static void main(String[] args) {

      Day birthday, deadline;
      birthday = new Day(2014,1,15);
      deadline = birthday;

      System.out.println(birthday); //15 Jan 2014
      System.out.println(deadline); //15 Jan 2014

      birthday.advance();
      System.out.println(birthday); //16 Jan 2014
      System.out.println(deadline); //_________________

      birthday.next();
      System.out.println(birthday); //_________________
      System.out.println(deadline); //_________________

      birthday = birthday.next();
      System.out.println(birthday); //_________________
      System.out.println(deadline); //_________________
}
```

```
public class Day {

  private int year;
  private int month;
  private int day;

  //Constructor
  public Day(int y, int m, int d) {
     this.year=y;
     this.month=m;
     this.day=d;
  }

  ..
  // create and return the "next day" of
  // the current day object
  public Day next() {..}

  // advance the current day object by 1 day
  public void advance() {..}
}
```

Q10. [Topic03- Page 7] Complete the blanks in the extended program:

```
class Day
{
   private int year;
   private int month;
   private int day;

   public Day(int y, int m, int d)
   {
      this.year=y;
      this.month=m;
      this.day=d;
   }

   public String toString()
   {
      final String[] MonthNames = {
                                "Jan", "Feb", "Mar", "Apr",
                                "May", "Jun", "Jul", "Aug",
                                "Sep", "Oct", "Nov", "Dec"};

      return day+" "+ MonthNames[month-1] + " "+ year;
   }
}

public class Main
{
   private static void testing1()
   {
      Day d1;
      System.out.println(d1);          ① __
      System.out.println(d1.toString()); ② __
   }

   private static void testing2()
   {
      Day d1 = null;
      System.out.println(d1);          ③ __
      System.out.println(d1.toString()); ④ __
   }

   private static void testing3()
   {
      Day d1 = new Day(2014,1,15);
      System.out.println(d1);          ⑤ __
      System.out.println(d1.toString()); ⑥ __
   }

   public static void main(String[] args)
   {
      testing_(); //Change this line to testing1(), testing2(), or testing3() for testing
   }
}
```

An exercise:

Consider this simplified Day class.

Your task: Match ① - ⑥ with the descriptions:

a. ***Error: d1 may not have been initialized***
[This is a compilation problem in Java.
If not fixed, we cannot run the program.]

b. ***Runtime exception: java.lang.NullPointerException***

c. ***print: 15 Jan 2014***

d. ***print: null***