

1. Java dynamic dispatch (20 points)

Given the following Java class definitions:

```
public class A {
    A() { System.out.println("A.A()"); }
    public void f(A arg) { System.out.println("A.f(A)"); }
    public void g(A arg) { System.out.println("A.g(A)"); }
};

public class B extends A {
    B() { System.out.println("B.B()"); }
    public void f(B arg) { System.out.println("B.f(B)"); }
    public void g(A arg) { System.out.println("B.g(A)"); }
    public static void h(A arg) { System.out.println("B.h(A)"); }
}

public class C extends B {
    C() { System.out.println("C.C()"); }
    public void f(A arg) { System.out.println("C.f(A)"); }
    public void f(B arg) { System.out.println("C.f(B)"); }
    public void g(B arg) { System.out.println("C.g(B)"); }
    public static void h(A arg) { System.out.println("C.h(A)"); }
}
```

What are the effects of each of the following statements executed in sequence (e.g., prints xxx, causes a run-time exception to be thrown, gives a compile-time error, etc.)?

- A a = new A();
prints A.A()
- A ab = new B();
prints A.A()
prints B.B()
- B bc = new C();
prints A.A()
prints B.B()
prints C.C()
- C c = new C();
prints A.A()
prints B.B()
prints C.C()
- a.f(bc);
prints A.f(A)
- bc.h(c)
prints C.h(a)
- bc.f(ab);
prints C.f(A)
- c.g(a);
prints B.g(A)
- c.g(bc);
prints C.g(B)
- ab.h(ab);
compile time error
- ab.g(ab);
prints B.g(A)

2. Java interfaces (20 points)

Given the following (partial) class definition:

```
public class Record implements Comparable {
    int i;
    String s;
    java.util.Date d;
    long l;

    public int compareTo(Object other) { }
}
```

Implement the **Record.compareTo()** method so that Records can be used in an ordered Collection (e.g., TreeSet). The ordering should be in ascending lexicographic, in the order the fields are declared (i.e. first by field **i**, then field **s**, etc.). Just to remind you, `this.compareTo(other)` should return a negative integer, zero, or a positive integer if *this* is less than, equal to, or greater than *other*. Assume that *other* will be an instance of Record.

```
public int compareTo(Object other) {
    // this could throw a ClassCastException, if other is not a Record
    Record r = (Record) other;

    if (i < r.i) return -1;
    if (i > r.i) return 1;

    int strCompare = s.compareTo(r.s);
    if (strCompare != 0) return strCompare;

    int dateCompare = d.compareTo(r.d);
    if (dateCompare != 0) return dateCompare;

    if (l < r.l) return -1;
    if (l > r.l) return 1;

    return 0;
}
```

3. Java Exceptions (20 points)

```
class Front extends ArrayIndexOutOfBoundsException {}

public class ExceptionFunction {
    public static boolean change( int[] course , int index ) {
        try { System.out.println("Start change");
            if (index < 0) { throw new Front(); }
            course[index] = 1;
        }
        catch (Front e) {
            course [-index] = 1; System.err.println("OutOfBounds: " + index);
            return false;
        }
        catch (ArrayIndexOutOfBoundsException e) {
            course[index - 7] = 1; System.err.println("OutOfBounds: " + index);
            return true;
        }
        finally {System.out.println("In Final Block"); }
        return false;
    }
}

public static void main( String args[] ) {
    try {int students[] = new int[5]; System.out.println("Main");
        System.out.println(change( students, Integer.parseInt(args[0])));
        System.out.println("After Change");
    }
    catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("Over Here");
    }
}
}
```

What is the output of this program when run as follows?

- java ExceptionFunction 2
Main
Start change
In Final Block
false
After Change
- java ExceptionFunction -2
Main
Start change
OutOfBounds: -2
In Final Block
false
After Change
- java ExceptionFunction 10
Main
Start change
OutOfBounds: 10
In Final Block
true
After Change
- java ExceptionFunction -10
Main
Start change
In Final Block
Over Here

4. Java Multithreading (25 points)

```
public class Auction {

    int currentBid;    int minBid;
    String highestBidder = "Nobody";
    boolean biddingOpen = true;

    Auction (int min) {minBid = min; currentBid = minBid;}

    synchronized void setBid (int bid, String offeredBy) {
//YOU MUST WRITE THIS CODE
// will accept bid if higher than currently offered bid
// bidders shouldn't be allowed to increase their own bids
// if bid accepted, bidder thread should block here until
// higher bid is accepted or bidder wins auction
    }

    synchronized int getBid () {return currentBid; }
    synchronized int getMinBid() {return minBid;}
    synchronized boolean biddingAllowed () {return biddingOpen;}
    synchronized void closeBidding () {biddingOpen = false;}
    synchronized String getHighestBidder() {
        String bidder = highestBidder; return highestBidder;
    }

    static public void main (String [] args) {
        Auction a = new Auction(100);

        Bidder b1 = new Bidder ("Bidder 1",a,X); // assume X, Y, and Z
        Bidder b2 = new Bidder ("Bidder 2",a,Y); // are defined
        Bidder b3 = new Bidder ("Bidder 3",a,Z);

        b1.start(); b2.start(); b3.start();
        while (a.biddingAllowed()) {

// Assume that this code checks once per second to see if
// the current highest bid has been raised. If three seconds
// pass without a new bid, auction goes to last highest bidder.
// You DON'T have to write any code for this part.
        }}} //end while //end main and //end Auction
```

```
public class Bidder extends Thread {

    String bidderName; Auction auction; int maxBid;

    Bidder (String name, Auction a, int top) {
        bidderName = name; auction = a; maxBid = top;
    }

    public void run () {
        int min; int currentBid = auction.getBid(); String currentBidder;

        //YOU MUST WRITE THIS CODE
        // If this bidder has been outbid, must try to raise bid by 1
        // up to maxBid
    }
}
```

Write the code for Auction's setBid() and for Bidder's run() methods. You may not change any code that is already written.

```

synchronized void setBid (int offered, String offeredBy) {
//ADD YOUR CODE HERE
// will accept bid if higher than currently offered
// bidder can't increase their own bid
// if bid accepted, bidder thread should block until
// higher bid is accepted or bidder wins auction
    boolean bidAccepted = false;
    if (offered > currentBid && offered > minBid &&
!highestBidder.equals(offeredBy)) {
        highestBidder = offeredBy;    currentBid = offered;
        bidAccepted = true;
        System.out.println("Bid of " + currentBid + "  accepted");
        notifyAll();
        try {wait();}
        catch (InterruptedException e) {};
    }
}

public void run () {
    int min;  int currentBid = auction.getBid();  String currentBidder;

    while (auction.biddingAllowed() && currentBid < maxBid) {

// ADD YOUR CODE HERE
// If bidder has been outbid, will try to raise bid by 1
// up to maximum bid
        synchronized (auction) {
            min = auction.getMinBid();
            currentBid = auction.getBid();
            currentBidder = auction.getHighestBidder();
            if (currentBid < maxBid) {auction.setBid(currentBid+1,bidderName);}
        }
    }
}

```

5. Java - short answers (15 points)

Answers should be no more than 4 sentences

(a) (5 points)

Why would a programmer implement the equals() method for a class? Give an example where the default implementation of equals() would not give the expected result.

If you wanted to compare for equality based on whether the fields in an object have the same contents, as opposed to referencing the same object. For example, String.equals() returns true if the characters in the String are the same, not whether two String references point to the same String object.

(b) (5 points)

Describe the differences between the four types of access control modifiers that Java provides to apply to class fields and methods.

public - any method can access the field or execute the method

private - only methods in the class

protected - only methods in the class or in derived classes

package - any method in the package of the class

(c) (5 points)

What are the effects of executing a synchronized block in Java.

locks object

refreshes read values

flushed written values