Question 1.

(a) /** An instance is for an out-of-space exception */

```java
public class OutOfSpaceException extends RuntimeException {
    /** Constructor: An instance with no detail message */
    public OutOfSpaceException() {
        super();
    }
    /** Constructor: An instance with detail message m */
    public OutOfSpaceException(String m) {
        super(m);
    }
}
```

(b) /** An instance maintains info about a hard drive. ... */

```java
public class HardDrive extends Storage {
    private int used; // Amount of spaced used on drive
    // Objects on the drive. The location of an object
    // is its index in this vector.
    private Vector contents;
    /** Constructor: empty hard drive with capacity c */
    public HardDrive(int c) {
        super(c);
        used= 0;
        contents= new Vector();
    }
    /** = amount of unused space */
    public int remainingSpace() {
        return getCapacity() - used;
    }
    /** Add ob, of size s, to drive, and return its location.
     * Throw OutOfSpaceException if not enough room */
    public int add(Object ob, int s) {
        return add(new MP3Song(ob, s), s);
    }
    /** = the title of song at location i */
    public String getTitle(int i) {
        MP3Song song= (MP3Song)(get(i));
        return song.title;
    }
    /** = the song at location i */
    public Object getSong(int i) {
        MP3Song song= (MP3Song)(get(i));
        return song.mp3;
    }
}
```

Question 2 (a)

```java
public MP3Song(Object f, String n) {
    if (!(n.endsWith(".mp3")))
        throw new IllegalArgumentException("title does not end in .mp3");
    mp3= f;
    title= n;
}
```

(b) /** Instance is a hard drive with only mp3 songs,
Each object on this drive is of class MP3Song. */

```java
public class MP3 extends HardDrive {
    /** Constructor: A new MP3 player with capacity c */
    public MP3(int c) {
        super(c);
    }
    /** Add song mp3, with title n and size s, to this MP3
    * drive and return its location.
    * Throw an IllegalArgumentException if ... */
    public int add(Object mp3, String n, int s) {
        return add(new MP3Song(mp3, n, s));
    }
    /** = the title of song at location i */
    public String getTitle(int i) {
        MP3Song song= (MP3Song)(get(i));
        return song.title;
    }
    /** = the song at location i */
    public Object getSong(int i) {
        MP3Song song= (MP3Song)(get(i));
        return song.mp3;
    }
}
```

Question 3.

```java
public static Point partition(int[] b, int h, int k) {
    int i= h;
    int j= i;
    int t= k;
    // inv:b[h..i-1 < x, b[i..j] = x,
    // b[j+1..t] is unknown, and b[t+1..k] > x
    while (j < t) {
        if (b[j+1] < b[i]) {
            j= j+1;
            int temp= b[j]; b[j]= b[i]; b[i]= temp;
            i= i+1;
        } else if (b[j+1] == b[i]) {
            j= j+1;
        } else {
            int temp= b[j+1]; b[j+1]= b[t]; b[t]= temp;
            t= t-1;
        }
    }
    return new Point(i, j);
}
```
**Question 4.**

```java
/** Place the m x n Bricks, as requested on the exam and return the array. */
public Brick[][] placeSquares(int m) {
    Brick[][] b = new Brick[m][m];
    for (int c = 0; c < m; c = c + 1) {
        // Place col c of bricks
        for (int r = 0; r < m; r = r + 1) {
            b[c][r] = new Brick(r * BrickSide, c * BrickSide, BrickSide);
            if (r == 0 || r == m - 1 || c == 0 || c == m - 1)
                b[c][r].setColor(Color.pink);
            else if (((r + c) % 2 == 0)
                b[c][r].setColor(Color.red);
            else b[c][r].setColor(Color.green);
            add(b[c][r]);
        }
    }
    return b;
}
```

**Question 5.**

```java
/** eqChars(String s, int i) { int k = 1; /** inv: Characters in s[i..i+k-1] are the same */
while (i+k < s.length() && s.charAt(i) == s.charAt(i+k)) {
    k = k+1;
}
return k; }
/** compress(String s) { if (s.length() == 0)
return s; // s has at least one character
int len = eqChars(s, 0);
return s.charAt(0) + (len + compress(s.substring(len)));
}
```

**Question 6.**

```java
(a) public interface P {
    int m(int x);
    void p();
}
(b) public class Answer extends JFrame implements P {
    public int m(int x) {
        return x;
    }
    public void p() {}
}
```

**Question 7.**

```java
(a) Evaluation the expression. If it is true, execute statement S1; otherwise, execute statement S2.
(b)  Alpha: 1
     a1
     n 5
     Alpha
     deal()
     f 10
     Alpha
     deal()
     f 12
     Beta
     deal()
```