Exercise Set: Control Flow and Data Models

A lot of the code snippets used in this exercise are taken from JHotDraw, an open-source project that makes it easy to write graphical editors, such as providing a drawing program. For more information on JHotDraw look at http://www.jhotdraw.org/. For more information on open source you can look at http://www.opensource.org/.

In this exercise set, we have marked questions we think are harder than others with a **[‡]**. We have also marked questions for which solutions are provided at the end of the set (**[SP]**). To check solutions for other questions than those marked with **[SP]**, ask one of the instructors or TAs or post a question to the google group!

1. Extract a flowchart from each of the following code snippets.

a)

```
public void promptOpen() {
	JFileChooser openDialog = createOpenFileChooser();
	if (openDialog.showOpenDialog(this) == JFileChooser.APPROVE_OPTION) {
		StorageFormat foundFormat =
			findStorageFormat(openDialog.getFileFilter());
		// if there is no format associated,
		// try to find one that supports the file
		if (foundFormat == null) {
			foundFormat =
				findStorageFormat(openDialog.getSelectedFile());
		}
		if (foundFormat != null) {
			loadDrawing(foundFormat);
		}
		else {
			showStatus("Not a valid file format: ");
		}
	}
}
```

b)

```
/**
 * Handles mouse down events and starts the corresponding tracker.
 */
public void mouseDown(MouseEvent e, int x, int y) {
	setView((DrawingView)e.getSource());
	if (e.getClickCount() == 2) {
		Figure figure = drawing().findFigure(e.getX(), e.getY());
		if (figure != null) {
			inspectFigure(figure);
			return;
		}
	}
	super.mouseDown(e, x, y);
}
```

c)

```
/**
 * Draws a pattern background pattern by replicating an image.
 */
private void drawPattern(Graphics g, Image image, DrawingView view) {
	int iwidth = image.getWidth(view);
	int iheight = image.getHeight(view);
	Dimension d = view.getSize();
	int x = 0;
	int y = 0;

	while (y < d.height) {
		while (x < d.width) {
			g.drawImage(image, x, y, view);
			x += iwidth;
		}
		y += iheight;
		x = 0;
	}
}
```

d) **[‡], [SP]**

```
/**
 * Find a StorageFormat that can be used according to a file object to store
 * a Drawing in a file or restore it from a file respectively.
 *
 * @param file a File object to be matched
 * @return StorageFormat, if a matching file extension could be found,
 * <code>null</code> otherwise
 */
public StorageFormat findStorageFormat(File file) {
	StorageFormat currentStorageFormat;
	for (int i=0; i<myStorageFormats.size(); i++) {
		currentStorageFormat = (StorageFormat) myStorageFormats.get(i);
		if (currentStorageFormat.getFileFilter().accept(file)) {
			return currentStorageFormat;
		}
	}
	return null;
}
```

2. Extracting call graphs:

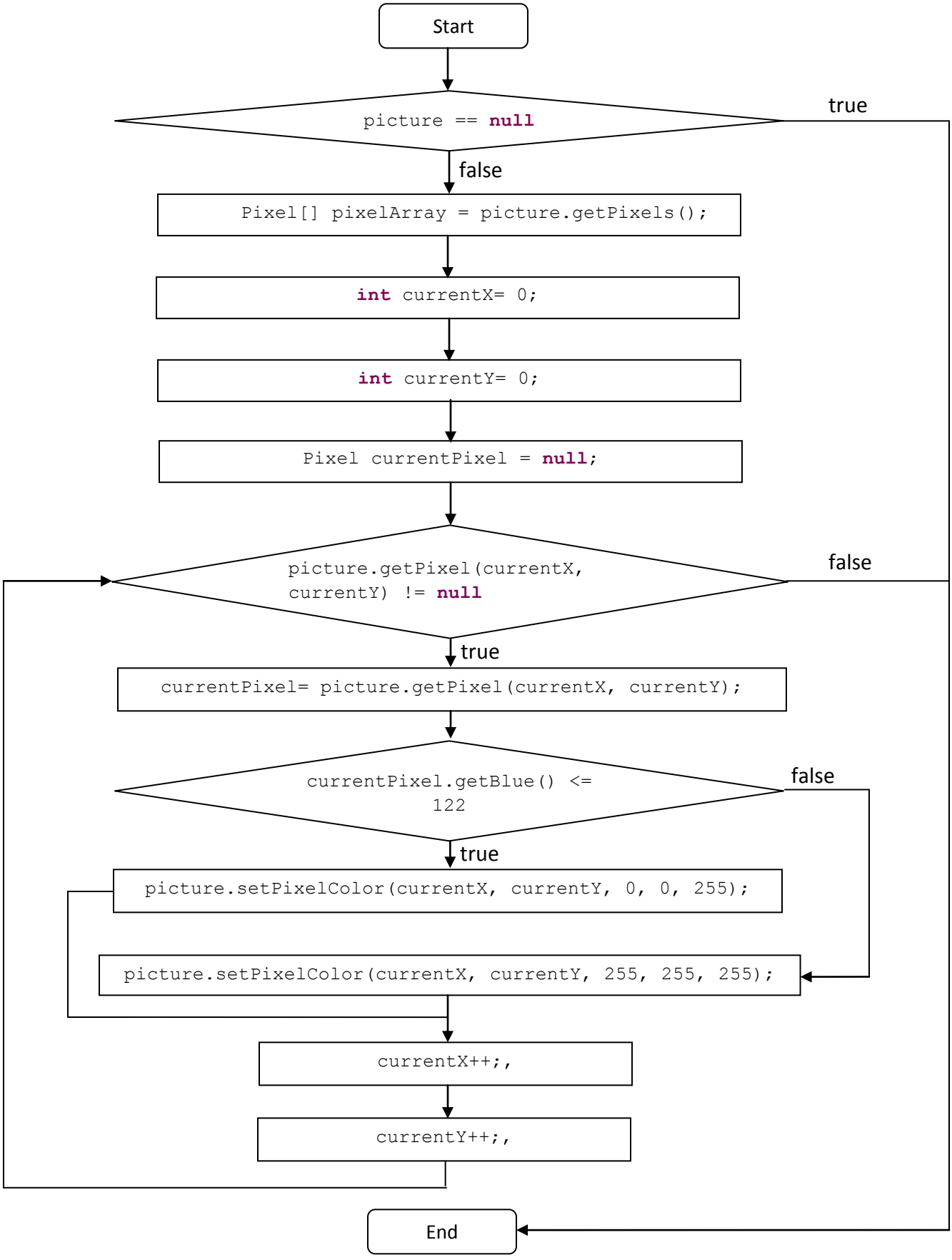
a) Extract a call graph starting from method `assignToTeams` from the class `TeamAssigner` (package `ca.ubc.cpsc.teammaker.actions`) in the TeamMaker system. Stop following method calls when a method is defined outside of the class TeamAssigner.

b) Extract a call graph starting from method `init` of class `JavaDrawViewer` (see below). Stop following method calls when a method is defined outside of the class JavaDrawViewer. **[SP]**

```
public class JavaDrawViewer extends JApplet implements DrawingEditor {

	private Drawing fDrawing;
	private Tool fTool;
	private StandardDrawingView fView;
	private transient UndoManager myUndoManager;

	public void init() {
		setUndoManager(new UndoManager());
		fView = new StandardDrawingView(this, 400, 370);
		setTool(new FollowURLTool(this, this));

		String filename = getParameter("Drawing");
		if (filename != null) {
			loadDrawing(filename);
			fView.setDrawing(fDrawing);
		}
		else {
			showStatus("Unable to load drawing");
		}
	}

	private void loadDrawing(String filename) {
		URL url = new URL(getCodeBase(), filename);
		InputStream stream = url.openStream();
		StorableInput reader = new StorableInput(stream);
		fDrawing = (Drawing)reader.readStorable();

	}

	protected Drawing createDrawing() {
		return new StandardDrawing();
	}

	/**
	 * Gets the editor's drawing view.
	 */
	public DrawingView view() {
		return fView;
	}

	/**
	 * Gets the editor's drawing.
	 */
```

```
	public Drawing drawing() {
		return fDrawing;
	}

	/**
	 * Gets the current the tool (there is only one):
	 */
	public Tool tool() {
		return fTool;
	}

	/**
	* Sets the current the tool
	*/
	public void setTool(Tool newTool) {
		fTool = newTool;
	}

	protected void setUndoManager(UndoManager newUndoManager) {
		myUndoManager = newUndoManager;
	}

	public UndoManager getUndoManager() {
		return myUndoManager;
	}
}
```

3. Given the following flowchart, explain in words what task the method accomplishes. **[‡]**



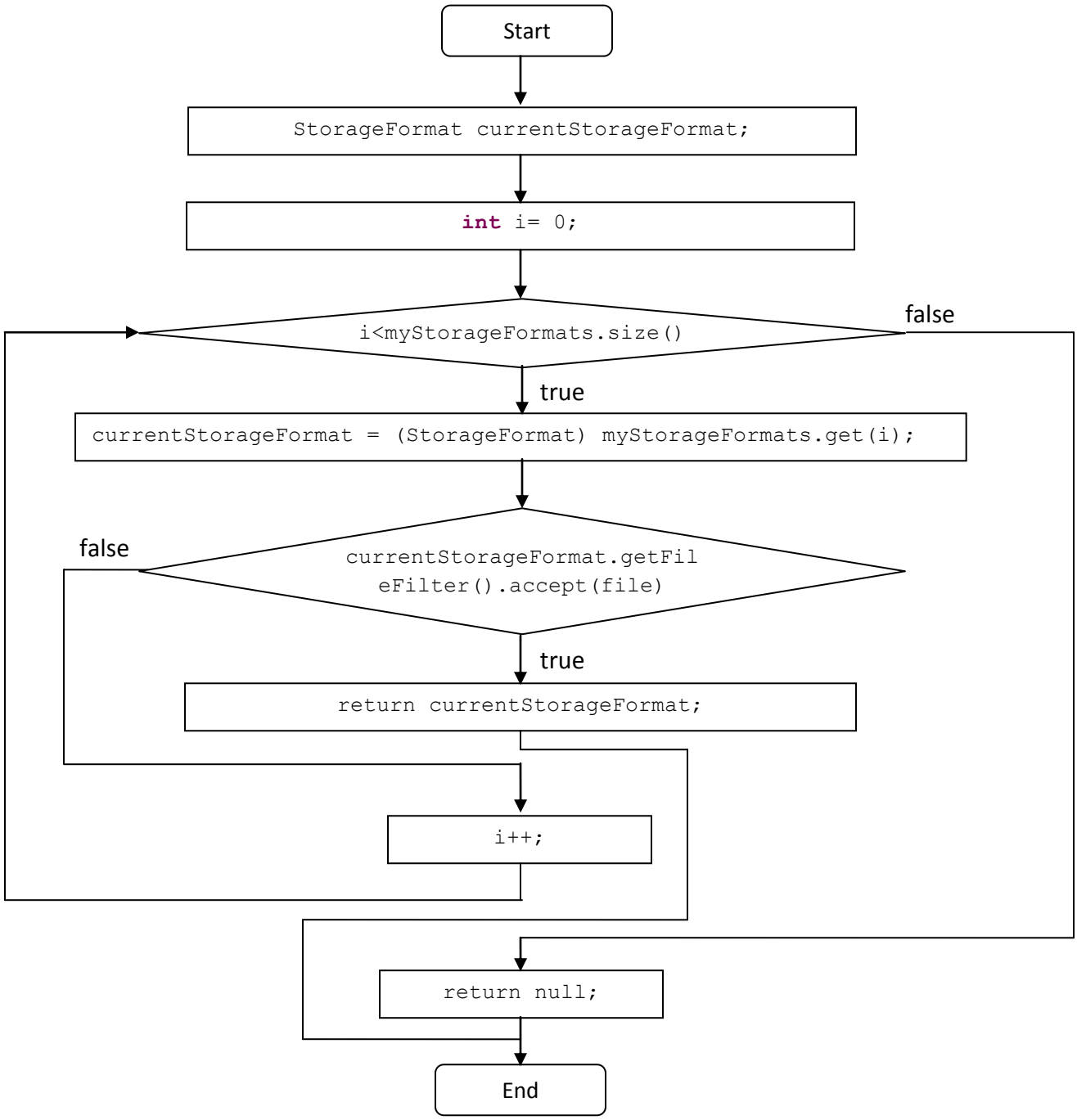
5. Given the following class representing a user in a chat forum (part of an open-source project), determine the essential data of the class.

```
public class User {
	// the internal id of the user
	private int id;

	// the user name
	private String username;

	// the password
	private String password;

	// the number of all posts from the user
	private int totalPosts;

	// the date the user last visited the chat forum
	private Date lastVisit;

	// if the user is still active this is set to 1
	private int active;

	// first name of the user
	private String firstName;

	// last name of the user
	private String lastName;

	// the combination of first and last name
	private String fullName;

	...
}
```

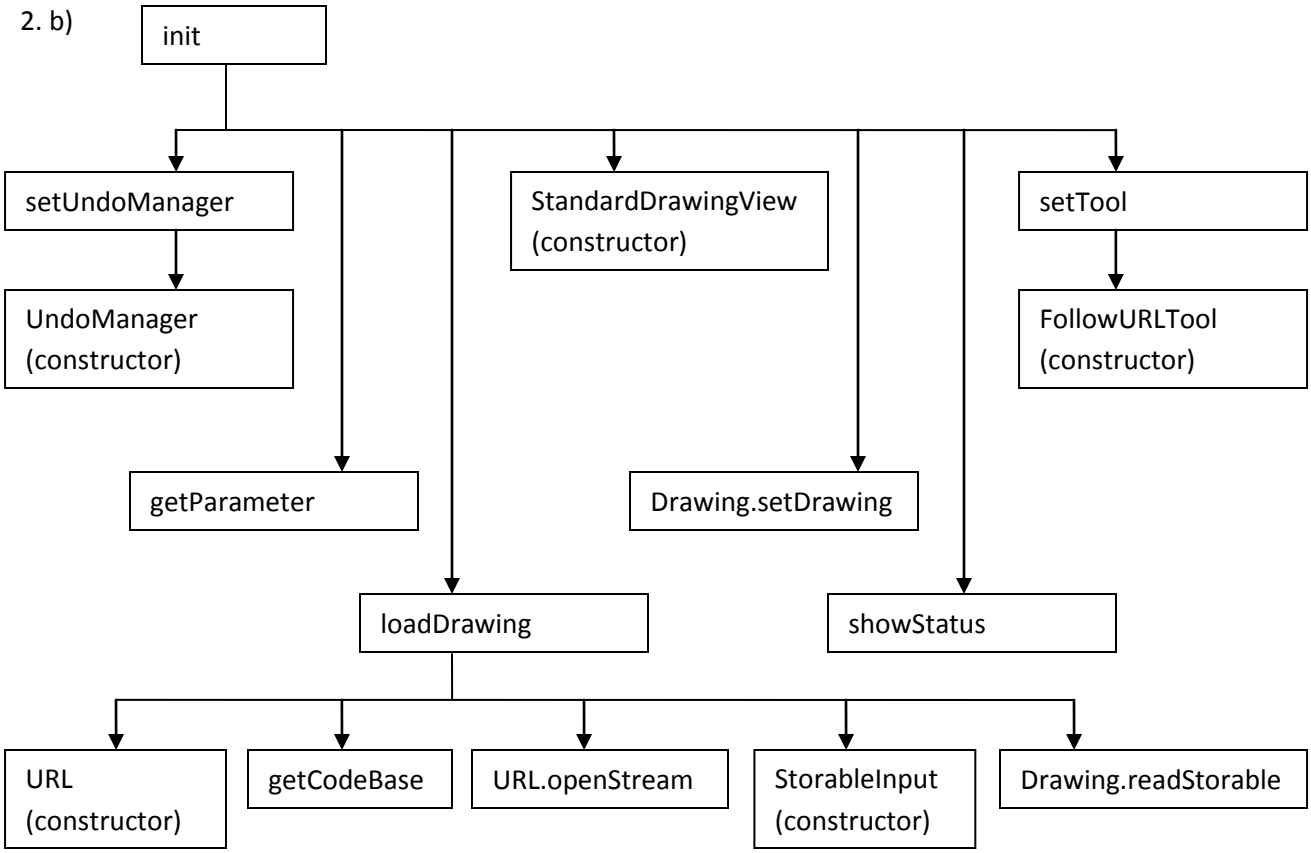
6. Assume you are about to develop an online application for selling books similar to Amazon. You are trying to come up with the data model for the application and have determined that there has to be at least the concept of a book and the one of a customer. In your opinion, what would be the essential data for a book class? What would be the essential data for a customer?

SOLUTIONS:

1. d)



2. b)


init
setUndoManager
StandardDrawingView
(constructor)
setTool
UndoManager
(constructor)
FollowURLTool
(constructor)
getParameter
Drawing.setDrawing
loadDrawing
showStatus
URL
(constructor)
getCodeBase
URL.openStream
StorableInput
(constructor)
Drawing.readStorable