Anexo:

```java
public interface IntegerElement {
    int value();
}
public class BSTImplementation<T extends IntegerElement> {
    private TreeRoot root;

    //...

    private class TreeRoot {
        T element;
        TreeRoot left, right;
    }
}
```

Fig 1

```java
public interface Vertex<V> {
    V element();
}
public interface Edge<E, V> {
    E element();
    Vertex<V>[] vertices();
}
public interface ValuedEdge {
    int value();
    boolean isActive();
}
public class GraphImplementation<V, E extends ValuedEdge> {
    //Estrutura de dados: lista de adjacências
    private Map<V, Vertex<V>> vertices = new HashMap<>();
    private Map<E, Edge<E, V>> edges = new HashMap<>();
       //...
}
```

Fig 2

```java
public interface Graph<V, E> {
   public int numVertices();
   public int numEdges();
   public Collection<Vertex<V>> vertices();
   public Collection<Edge<E, V>> edges();
   public Collection<Edge<E, V>> incidentEdges(Vertex<V> v)
         throws InvalidVertexException;
  public Vertex<V> opposite(Vertex<V> v, Edge<E, V> e)
   public boolean areAdjacent(Vertex<V> u, Vertex<V> v)
         throws InvalidVertexException;
   public Vertex<V> insertVertex(V vElement)
         throws InvalidVertexException;
   public Edge<E, V> insertEdge(Vertex<V> u, Vertex<V> v, E edgeElement)
         throws InvalidVertexException, InvalidEdgeException;
   public Edge<E, V> insertEdge(V vElement1, V vElement2, E edgeElement)
         throws InvalidVertexException, InvalidEdgeException;;
   public V removeVertex(Vertex<V> v) throws InvalidVertexException;
   public E removeEdge(Edge<E, V> e) throws InvalidEdgeException;
   public V replace(Vertex<V> v, V newElement) throws InvalidVertexException;
   public E replace(Edge<E, V> e, E newElement) throws InvalidEdgeException;
}
```

Fig3

```java
public class SocialNetwork {
   private Graph<Person,Date> network;

 /** construtor and methods**/
```

Fig 4

```java
public class Team extends Observable implements Iterable<String> {

    private final List<String> persons;
    private String name;

    public Team(String name) {
        this.name = name;
        persons = new ArrayList<>();
    }
    public void addWord(String s) throws TeamException {
        if (persons.contains(s)) {
            throw new TeamException(String.format("A palavra %s já existe!", s));
        }
        persons.add(s);
        setChanged();
        notifyObservers();
    }
    public void clear() {
        persons.clear();
        setChanged();
        notifyObservers();
    }
    public int getWordCount() {
        return persons.size();
    }

    @Override
    public String toString() {
        return "Team{" + persons + ", name=" + name + '}';
    }
    @Override
    public Iterator<String> iterator() {
        return persons.iterator();
    }
}
```

```java
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) {

        Team team = new Team("Hoquei 1");
        TeamUI ui = new TeamUI(team);
        TeamController controller = new TeamController(ui, team);
        BorderPane window = new BorderPane();
        window.setCenter(ui);
        Scene scene = new Scene(window, 300, 250);
        primaryStage.setTitle("Team Builder");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

}
```

```java
public class TeamUI extends VBox implements Observer{

    //controlos
    private TextField txtInputWord;
    private Button btAddWord;
    private Button btUndo;
    private ListView<String> listWords;
    private Label lblCount;
    private final Team team;

    public TeamUI(Team team) {
        this.team = team;
        initComponents();
        update(team, null);
    }

    @Override
    public void update(Observable o, Object o1) {
        if(o instanceof Team) {
            Team model = (Team)o;
            listWords.getItems().clear();
            for(String word:model)
                listWords.getItems().add(word);
            lblCount.setText(""+ model.getWordCount());
        }
    }
    private void initComponents()      {
        txtInputWord = new TextField();
        btAddWord = new Button("Add");
        listWords = new ListView<>();
        lblCount = new Label("0");
        HBox firstRow = new HBox(txtInputWord, btAddWord,lblCount);
        firstRow.setSpacing(5);
        this.getChildren().addAll(firstRow, listWords);
    }

    public void setTriggers(TeamController controller) {
        btAddWord.setOnAction((ActionEvent event) -> {
            controller.doAddWord();
        });
    }

    public String getInputWord() {
        return txtInputWord.getText().trim();
    }

    public void clearInput() {
        txtInputWord.setText("");
    }
```

Fig 5