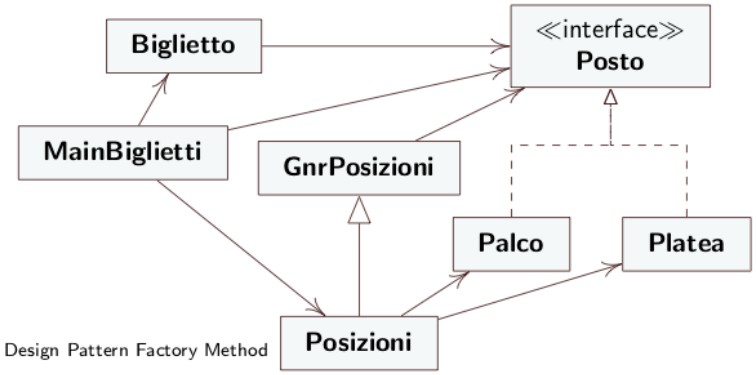
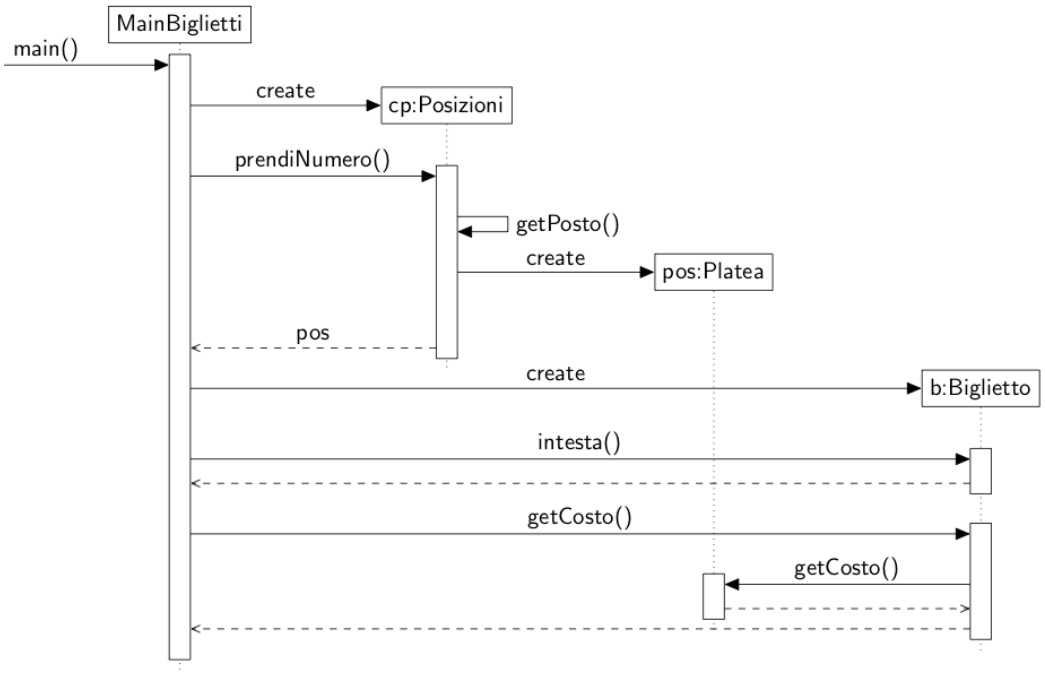


Design Pattern Factory Method




```java
// Codice Java che implementa il design pattern Factory Method

// Posto e' un Product
public interface Posto {
   public String getPosizione();
   public int getCosto();
   public String getSettore();
}


// Palco e' un ConcreteProduct
import java.util.Random;

public class Palco implements Posto {
   private final int numero;

   public Palco() {
      numero = new Random().nextInt(20) + 1;
   }

   @Override
   public int getCosto() {
      if (numero > 10) return 50;
```

```
        return 40;
    }

    @Override
    public String getPosizione() {
        return Integer.toString(numero);
    }

    @Override
    public String getSettore() {
        if (numero == 20) return "Centrale";
        if (numero > 10) return "Verde";
        return "Blu";
    }
}


// Platea e' un ConcreteProduct
import java.util.Random;

public class Platea implements Posto {
    private final String[] nomi = { "A", "B", "C", "D", "E", "F" };
    private final int numero;
    private final int riga;

    public Platea() {
        numero = new Random().nextInt(10) + 1;
        riga = new Random().nextInt(5) + 1;
    }

    @Override
    public int getCosto() {
        if (numero > 5 && rigaMax()) return 100;
        if (numero > 5 && rigaMin()) return 80;
        return 60;
    }

    @Override
    public String getPosizione() {
        return nomi[riga].concat(Integer.toString(numero));
    }

    @Override
    public String getSettore() {
        if (riga == 0) return "Riservato";
        return "Normale";
    }

    private boolean rigaMax() {
        return (riga >= 1 && riga <= 4);
    }
```

```
   private boolean rigaMin() {
      return (riga == 0 || riga == 5);
   }
}


// GnrPosizioni e' un Creator
import java.util.Set;
import java.util.TreeSet;
public abstract class GnrPosizioni { // versione 1.1
   private static final int MAXPOSTI = 100;
   private final Set< String > pst = new TreeSet< >(); // set di posti

   // metodo che rimanda alla sottoclasse l'istanziazione della classe
   public Posto prendiNumero(int x) {
      if (pst.size() == MAXPOSTI) return null;
      // il chiamante dovrebbe controllare se null
      Posto p;
      do {                // fino a quando non trova un posto libero
         p = getPosto(x); // chiama metodo della sottoclasse
      } while (pst.contains(p.getPosizione()));
      pst.add(p.getPosizione());
      return p;
   }

   public void printPostiOccupati() {
      for (String s : pst)
         System.out.print(s + " ");
   }

   // il metodo factory e' dichiarato ma non implementato
   public abstract Posto getPosto(int tipo);
}


// Posizioni e' un ConcreteCreator con un metodo factory
public class Posizioni extends GnrPosizioni {
   // metodo factory che ritorna una istanza
   @Override
   public Posto getPosto(int tipo) {
      // crea l'istanza di un ConcreteProduct
      if (1 == tipo) return new Palco();
      return new Platea();
   }
}


// Biglietto e' un client del Product Posto
public class Biglietto {
   private String nome;
   private final Posto pos;

   public Biglietto(Posto p) {
```

```
        pos = p;
    }

    public void intesta(String s) {
        nome = s;
    }

    public String getDettagli() {
        return nome.concat(" ").concat(pos.getPosizione());
    }

    public String getNome() {
        return nome;
    }

    public int getCosto() {
        return pos.getCosto();
    }
}


// Classe con il main che usa il ConcreteCreator
public class MainBiglietti {
    private static Posizioni cp = new Posizioni();

    public static void main(String[] args) {
        Posto pos = cp.prendiNumero(0);
        Biglietto b = new Biglietto(pos);
        b.intesta("Mario");
        System.out.println("Costo " + b.getCosto());

        new Biglietto(cp.prendiNumero(0));
        new Biglietto(cp.prendiNumero(0));
        cp.printPostiOccupati();
    }
}
```