1)

1. Критические секции, события, потоки, дескрипторы

2. Поток - объект ядра, которому операционная система выделяет процессорное время для выполнения приложения.

3. Мьютекс(mutex, mutual exclusion) - объект ядра операционной системы, предназначенный для синхронизации потоков. Может принадлежать только одному потоку одновременно, находится в сигнальном состоянии если не принадлежит ни одному потоку и иначе в несигнальном.

4. Событие - объект для оповещения одним потоком другой поток о выполненных действиях.

2)

1. ООП декомпозиция - разбиение задачи на более мелкие компоненты, которые могут быть реализованы в виде классов. Разделение ответственности и функциональности между компонентами и классами.

2. Статический полиморфизм - вил полиморфизма, при котором используется одно и то же имя метода или оператор для работы с разными типами данных. При статическом полиморфизме уже на этапе компиляции известно, какие конкретно будут вызваны методы (т.к. в этом случае на уровне компиляции уже известно с какими типами мы работаем). Реализуется с помощью перегрузки методов и операторов.

3. Инкапсуляция - принцип ООП, который требует объединения связанный между собой методов и данных в одной сущности, а также скрытия внутренней реализации, предоставляется только интерфейс для работы с данными.

3)

1. Builder - шаблон проектирования, использующийся для упрощения создания сложных объектов поэтапно. На каждом этапе объект строится отдельным методом и в итоге формирование заканчивается методом build, который возвращает готовый объект.

public class Cake {

private String doughType;

private bool isGlazed;

public Cake(String \_type, bool \_glazed) {

this.doughType = \_type;

this.isGlazed = \_glazed;

}

// other methods

}

public class CakeBuilder {

private String doughType;

private bool isGlazed;

public CakeBuilder setDoughType(String \_type) {

this.doughType = \_type;

return this;

}

public CakeBuilder setGlazed(bool \_glazed) {

this.isGlazed = \_glazed;

return this;

}

public Cake build() {

return new Cake(doughType, isGlazed);

}

}

public class Director {

private CakeBuilder builder;

public Director(CakeBuilder builder) {

this.builder = builder;

}

public Cake construct() {

return builder.setDoughType("bisque")

.setGlazed(true)

.build();

}

}

public class Main {

public static void main(String[] args) {

CakeBuilder builder = new CakeBuilder();

Director director = new Director(builder);

Cake cake = director.construct();

}

}

2. Decorator – шаблон проектирования, который позволяет динамически добавлять новую функциональность объекту. Для этого он оборачивается в объект-декоратор.