Антипаттерны.
Объем написанного мной кода относительно небольшой, однако вот что удалось обнаружить:

Антипаттерны в коде:

1. Magic Numbers: единичные случаи в простых примерах
2. Спагетти-код: открыл проект, где, как мне казалось, такой код был. Но нет – теперь всё выглядит понятно и логично ☺. Пока с таким кодом не сталкивался, т.к. ещё имею небольшой опыт.
3. Lasagna code: пока не встречал.
4. Blind faith: есть. В месте, где требуется ввод имени пользователя в проекте по сетевому чату. Отдельно на лекции было сказано про SQLInjection. И эту проблему устранили, используя PreparedStatement вместо Statement. В поле для ввода пароля также применяли регулярные выражения. Это считается как защита от этого антипаттерна?
5. Cryptic code: использую часто. Буду использовать имена, раскрывающие смысл сущностей.
6. Hard code: использую часто, особенно для разных путей и имен файлов. Буду более широко использовать переменные среды и конфигурационные файлы.
7. Soft code: пока не обнаружил.
8. Lava flow: пока не обнаружил.

Антипаттерны в ООП:

1. Anemic Domain Model: пока не обнаружил
2. God object: встречается. Есть большие классы с явно избыточной функциональностью. Хотя стоит ли их упрощать – тоже большой вопрос, т.к. проект, где я это нашел, небольшой. Здравый смысл подсказывает оставить всё как есть в этом конкретном случае.
3. Poltergeist: не обнаружил
4. Singletonitis: пока не обнаружил
5. Privatization: попадается. Решение: важные методы в классе лучше объявлять protected, чтобы иметь возможность их переопределять в потомках (если это не final класс)
6. Interface soup: не встречал
7. Stub: не обнаружил

Методологические антипаттерны:

1. Copy-paste: даже если и нахожу, как мне кажется, подходящее решение, то перепечатываю его и пытаюсь разобраться, а не просто копипастю его напрямую.
2. Golden hammer: не нашел, но замечаю, что в первую очередь рассматриваю решения, которые мне понятны и знакомы, чем те, которые могут быть наиболее эффективны для решения проблемы, но в которых надо дополнительно разобраться. Хмм… Буду проводить поиск подходящего решения объективно, а не выбирая из своего инструментария.
3. Improbability factor: встречается. Возвращаемые результаты не всегда проверяю не наличие ошибок. Решение: обязательно проверять возвращаемые результаты.
4. Premature optimization: не встречал. Стараюсь сначала сделать работающий код, потом оптимизирую.
5. Reinventing the weel: Господи, благослови базы знаний для разработчиков!
6. Reinventing the square wheel: как и с golden hammer, иногда выбор в пользу решения делаю не объективно, а с учетом того, что знаю/умею. Решение здесь аналогично.

Архитектурные антипаттерны:

1. Abstract inversion: пока не встречал
2. Big ball of mud: монолиты можно отнести сюда? Если да, то одно из решений – использовать микросервисную архитектуру, если это приемлемо.
3. Input kludge: встречалось. Решение: использовать спецификации на ввод данных, например, регулярные выражения, проверять и обезвреживать введенные данные.
4. Magic button: пока не встречал.
5. Mutilation: пока не встречал.